THE UNIVERSITY OF HULL

Cartographic Information Systems Research Group
C.I.S.R.G.

Hon. Co-ordinator:
Dr. M. Visvalingam
Department of Computer Science
The University
HULL HU6 7RX

Tel. (0482) 465295/465951
Telex 592592 KHMAIL G,
f.a.o. HULIB375
Fax (0482) 466666

Discussion Paper Series Editors : Dr M Visvalingam
Dr M E Turner

CISRG DISCUSSION PAPER 4

The Douglas-Peucker Line Simplification Algorithm:

an introduction with programs

J D Whyatt and P R Wade

Paper submitted for publication in the Bulletin of the Society of University Cartographers

August 1988

List of CISRG reports:

Discussion Papers

1. Wade, P., Visvalingam, M. and Kirby, G.H. (1986)
   From Line Geometry to Area Topology, 48 pp.

2. Visvalingam, M. (1987)
   Problems in the Design and Implementation of a GKS-based User Interface for a Graphical Information System, 18 pp.

3. Visvalingam, M. (1988)
   Standardising Basic Spatial Units : Problems and Prospects, 19 pp.

4. Whyatt, J. D. and Wade, P. R. (1988)
   The Douglas-Peucker Algorithm for Line Simplification: an introduction with programs, 17 pp.

Special Issues

Visvalingam, M. (ed) (1986)
Research based on Ordnance Survey Small-Scales Digital Data
Proceedings of a meeting held as part of the 1986 Annual Conference of the Institute of British Geographers (Reading, Jan 1986), **sponsored by the Ordnance Survey,** 79 pp.

Visvalingam, M. and Kirby, G. H. (1987)
Directory of Research and Development based on Ordnance Survey Small Scales Digital Data, **sponsored by the Ordnance Survey,** 38 pp.

CONTENTS

ABSTRACT

The aim of this paper is to introduce the reader to one theme in digital cartography, namely line simplification. The paper describes the technique developed by David Douglas and Thomas Peucker in 1973. The reader is encouraged to experiment with data of their own choice using software supplied in the appendices, and provide feedback on the performance of this algorithm.

1. Introduction

Robinson, Sale and Morrison (1978) described four processes which they term 'The Elements of Generalisation'. These processes, namely simplification, classification, symbolisation and induction were considered important transformations in ensuring that information displayed on the map appeared in a clear, concise and uncluttered form. Most of the research effort on automated generalisation has been directed towards the simplification of line information.

Line information from a map is converted into computer readable numeric form by the use of a coordinate digitiser, raster scanner or automatic line following device. A line is thus stored in the computer as a series of representative points. Digitising techniques invariably record lines with far more detail than is necessary for accurate graphic reproduction or computer analysis, hence captured data is normally 'weeded' prior to being stored on a permanent basis. Weeding removes data representing unnecessary points, such as duplicates or points captured along a straight line. Weeding thus reduces data storage requirements and also increases the speed at which data may be retrieved from disc and plotted.

Even after weeding, there remains a need for scale related simplification. As a general rule, the smaller the map scale, the greater the level of simplification required. At each scale there is a level of simplification which results in the data being depicted

appropriately in neither too detailed nor too general a form. Displaying the British coastline at a scale of 1:1 250 000 will necessitate the removal of much more line information than depiction of the same information at a scale of 1:625 000. In the context of scale related simplification, detail should thus be omitted if it is not going to be visible at the scale at which the map is being displayed.

The scale at which a map may be displayed is dependent upon the resolution of the nominated output device. Carter (1984) provides a good introduction to raster and vector output devices. Both screen displays and dot matrix printer plotters are raster devices. In raster format a picture is formed by a matrix of dots, known as picture elements or pixels. Line detail finer than a pixel is therefore visually redundant, and should not be included in the data for displaying the line for reasons of efficiency. On a vector output device a line is portrayed by a series of straight line segments which connect the digitally captured points. A good vector plotter has a resolution of 0.005 inch. The highest quality plotters have resolution of 0.0001 inch. In vector graphics, the amount of line detail depicted is not limited by the resolution of the output device, but by the pen width selected by the user. Detail finer than the selected pen width becomes redundant, hence the requirement to simplify the digital data in order to remove such redundancy.

To summarise, there is a need to clean and simplify linear information prior to display. The level of simplification necessary is dependent upon the nominated output device and the scale of display. Linear simplification may thus be defined as 'the ' elimination of unwanted detail', in relation to these factors.

The Douglas-Peucker line simplification algorithm has been widely used over many years for data cleaning and simplification. This algorithm has also been used as a scale independent method for generalising line features, including boundaries in thematic mapping. White (1983) attempted to perceptually evaluate three line

generalisation algorithms. Her results showed that the Douglas-Peucker algorithm produced generalisations which appeared most similar to lines generalised by manual generalisation techniques. McMaster (1987) produced a series of geometric measures to evaluate the changes produced by line simplification. Of the nine algorithms evaluated, McMaster concluded that the Douglas-Peucker algorithm was 'mathematically superior'.

The Douglas-Peucker algorithm has also been put to a number of secondary uses other than simplification in recent years. Buttenfield (1986) attempted to identify various geomorphic features such as fjords and submerged coastlines using metrics derived from the Douglas-Peucker method of tolerancing. Williams (1987) developed two algorithms which maintained the relative areas of polygons after the boundaries had been generalised using the Douglas-Peucker algorithm. Jones and Abraham (1987) used the Douglas-Peucker algorithm to store linear features at levels of scale related significance within a scale-independent database.

From the above discussion it becomes clear that the Douglas-Peucker algorithm is a widely used method for line simplification and generalisation. This paper describes the algorithm, and three implementations of it, in detail. The reader is encouraged to experiment with the software provided using data of their own choice and comment upon the performance of the algorithm in cartographic terms. We believe that the Douglas-Peucker method has several shortcomings, and therefore would appreciate comments from others.

2. Description of the Algorithm

In their original paper (Douglas and Peucker, 1973), the authors describe two methods for reducing the number of points required to represent a digitised line. The second method has been most widely implemented; hence we shall now describe this in some detail.

The first point on the line is defined as the 'anchor' and the last point as a 'floater'. These two points are connected by a straight line segment and perpendicular distances from this segment to all intervening points are calculated. If none of these perpendicular distances exceed a user specified tolerance (distance value), then the straight line segment is deemed suitable to represent the whole line in simplified form.

If this condition is not met, then the point with the greatest perpendicular offset from the straight line segment is selected as a new floating point. The cycle is repeated, the new straight line segment being defined by the anchor and the new floater. Offsets for intervening points are then recalculated perpendicular to this new segment. This process continues; the line being repeatedly subdivided with selected floating points being stored in a stack, until the tolerance criteria is met. Once the tolerance criteria has been met, the anchor is moved to the most recently selected floater, and the new floating point is selected from the top of the stack of previously selected floaters.

The selection process is repeated. Eventually, the anchor point reaches the last point on the line, and the simplification process is complete. Points previously assigned as anchors are connected by straight line segments to form the simplified line. (A worked example of this method is shown in APPENDIX 1). Note that specifying a low tolerance value results in little line detail being removed whereas specifying a high tolerance value results in all but the most general features of the line being removed.

3. Software

In its original form, the Douglas-Peucker algorithm either selects or omits points according to the level of tolerance specified. All that is returned is a series of plotting coordinates. Thus, if a user wishes to observe the effect of different tolerance values, the process has to be repeated for each value.

GIMMS, a mapping package developed by Dr. T. C. Waugh at Edinburgh University, contains two commands for the simplification of line information using the Douglas-Peucker algorithm (See GIMMS Reference Manual p 5.5 - 5.7, Waugh and McCalden, 1983).

The simplest command to use is the REDUCE command. Supplying the command REDUCE=50 in the FILEIN module of a GIMMS program causes the removal of all points within a 50 unit tolerance of the trend of the line. Figure 1 depicts a sample line prior to simplification. Figure 2 depicts the line in its simplified form, having used the GIMMS command REDUCE=1000. APPENDIX 2 lists the program which created this simplification. The diagnostics file generated on every occasion a GIMMS program is run contains information concerning the number of points which have been used to create the simplified line.

The second command is slightly more complicated to use. The GENERAL command allows 'generalisation codes' to be attached to the coordinates in polygon files. The codes range in value from 0 to 9. End points of lines are assigned codes of 0. The GENERAL command is used to specify the offset values which correspond to the generalisation codes in order of decreasing value. Up to 9 offset values may be supplied. The generalisation code for the first (largest) offset is 1, with codes increasing by 1 for each subsequent offset given. The GENERAL command uses the Douglas-Peucker algorithm as described earlier in this paper. On each occasion the line is subdivided at the point of maximum offset, a code is computed for this point. The offset of this selected point is tested against the first (largest) input value specified in the GENERAL= statement. If its offset value exceeds this input value, then a code of 1 is stored with this point. If the offset is less than this value, it is tested against the next, slightly smaller, input value. This latter process repeats until the point is assigned a code.

The *DRAWMAP command (see APPENDIX 3) specifies the amount of detail that will be shown when the map is drawn. *DRAWMAP GENERAL=3 would result in the map being produced using points with generalisation

codes of three or less. The lower the GENERAL level specified, the more simplified the output appears. *DRAWMAP GENERAL=0 results in only the end points of lines being used to produce the map.

Figures 3a-d have been produced using GENERAL values of 50, 25 and 12 map units. The reader may retrieve the generalisation codes for each point by using the GIMMS *FILEDUMP facility (See GIMMS Reference Manual p 9.5).

This implementation effectively allocates the point data into classes as specified by the user. Experimentation with GENERAL levels necessitates the recalculation of codes on each occasion the data is output in graphic form. This may be regarded as being inefficient. However, plotting time is greatly reduced once the codes have been calculated. By specifying class intervals, this method of simplification becomes scale dependent. The GENERAL values in the *DRAWMAP command would need altering if the data were to be depicted at different scales.

In APPENDIX 4 we present our implementation of the Douglas-Peucker algorithm in Fortran 77 for an ICL 3980 computer running the VME operating system. The user is prompted to specify a tolerance value, and all points which exceed this value are returned by the program, along with their offset values. The latter may easily be omitted if the data is to be read straight into a plotting routine. If a tolerance of zero is specified, all points, with their associated offset values are returned. A separate filtering program is used to read this file. The user is therefore capable of retrieving data at any level of simplification without having to recompute offset values. Note that in the event of the calculated offset of a point exceeding that of the previously selected point, the offset value of the previously selected point is output in our implementation. In the GIMMS GENERAL implementation, different codes would be output for the two points if the difference between their offsets proved significant.

The original Douglas-Peucker method only allows the selection or omission of points. The GENERAL command in the GIMMS implementation allows the class ranking of points according to their perceived importance. Since our method returns actual offset values, it is possible to compare the relative importance of points in quantitative terms. Such information is valuable for evaluation purposes.

4. Data

If the reader wishes to use either of the GIMMS routines, then the data must be retrieved in either GIMMS line or GIMMS segment format. The REDUCE command will operate on either line or segment data, whereas the GENERAL command may only be used to simplify segment data.

Data may be obtained by either:

i. Digitising a base map.
ii. Retrieving data from a cartographic database.

The latter option is recommended for two reasons. Firstly, data may be retrieved very rapidly from a cartographic database; digitising is a slow process in comparison. Secondly, data held in a database has usually been weeded. If the reader manually digitises a map, then the data will need weeding prior to being used in a generalisation routine. Suitable data may be held at Regional Computer Centres. For instance, the South West Universities Regional Computer Centre (SWURCC) holds boundary data captured by the Department of the Environment and the Scottish Development Department. This data was originally digitised from Ordnance Survey or Bartholomews Post Office maps, mostly at a scale of 1:50000. Readers interested in using this data should contact Mr S M Wise at SWURCC (email wise@uk.ac.swurcc) for further details.

Our implementation of the Douglas-Peucker algorithm requires data supplied in the format listed in APPENDIX 5. We recommend retrieval of data from a database in GIMMS line format, and subsequent editing into the required form.

For further information concerning other sources of data, the reader should refer to the OS Directory of Research and Development (Visvalingam and Kirby, 1987).

Fig3a.

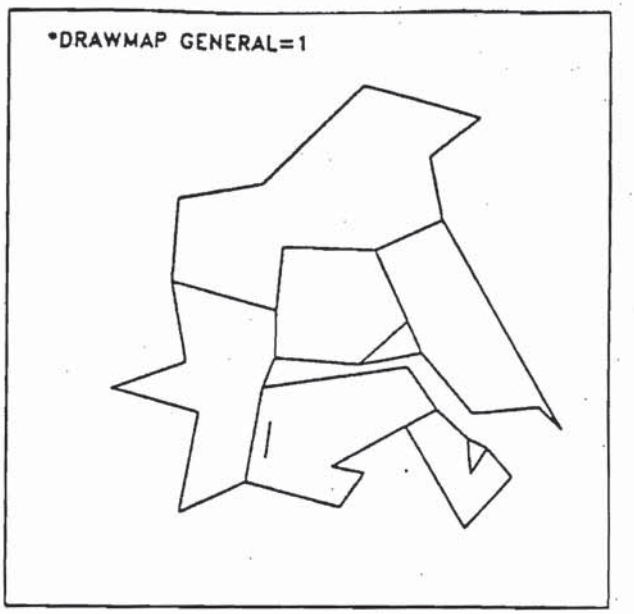

Fig3b.

Fig3c

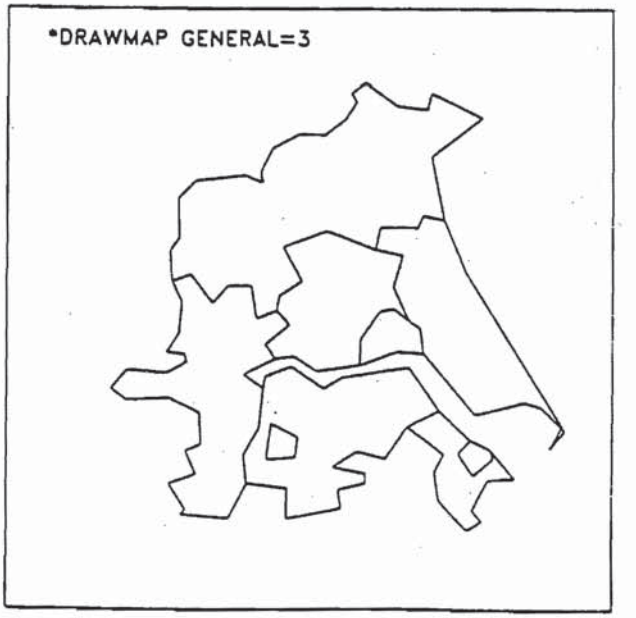

Fig3d

Figures 3a-d . Simplification using the GIMMS GENERAL command

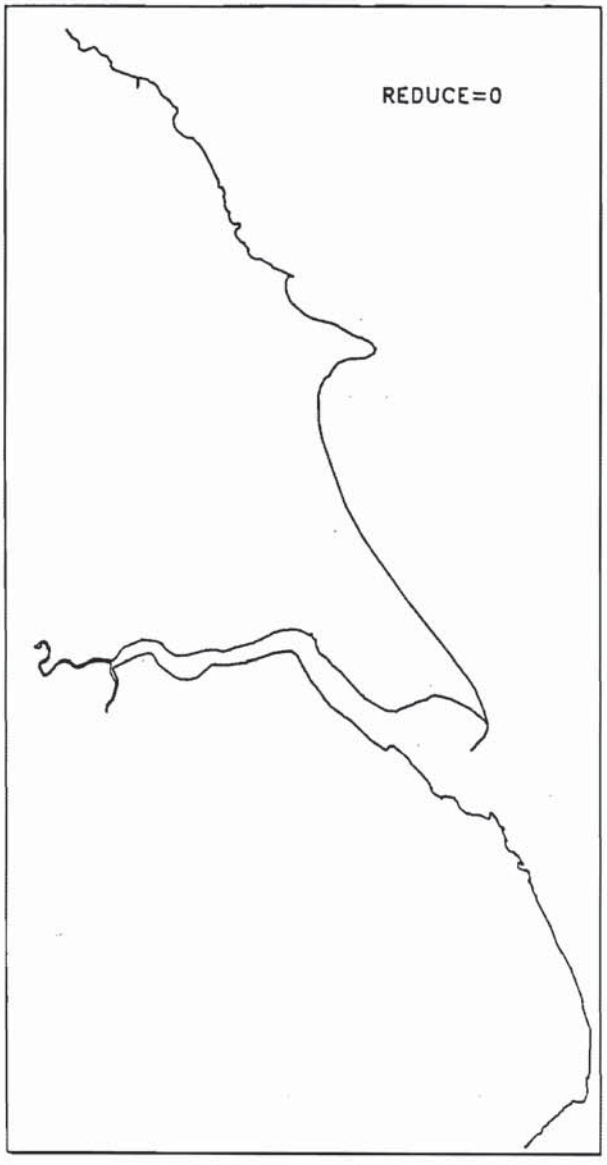

Figure 1. Humberside coastline prior to simplification

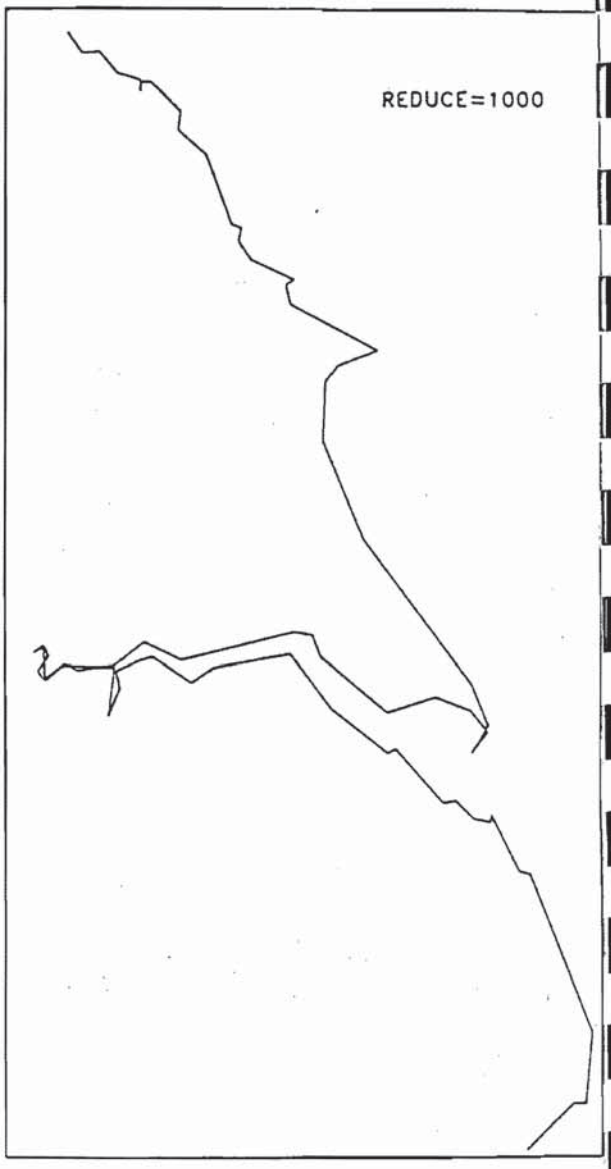

Figure 2. Humberside coastline after simplification

5. Conclusions

The Douglas-Peucker algorithm has been widely used for simplification and generalisation for many years. We believe that it produces excellent results under certain conditions, and is a good algorithm for weeding data. However, used as a line simplification algorithm, we feel that this technique has several shortcomings, particularly in the context of scale independent simplification. Our own observations have shown that the algorithm often produces unbalanced simplifications, the tendency being to oversimplify smooth sections of a line whilst retaining far too much detail in other areas. A more detailed account of our observations will be presented in a separate paper.

Other workers have also expressed dissatisfaction with the Douglas-Peucker algorithm. In the context of scale dependent simplification, Monmonier (1986) stated that the Douglas-Peucker algorithm was "successful only where scale reduction was comparatively minor and the density of features was sufficiently low so that overlap, or worse, criss-crossing tended not to occur". Thapa (1988) discussed the inapplicability of the Douglas-Peucker algorithm in terms of line simplification when the scale change between an original map and a generalised map was drastic. The author stated that in such instances the algorithm left spikes, and the simplification produced appeared cluttered and unclear.

We hope that the information presented in this paper will encourage others, including those with little computing knowledge, to experiment with the Douglas-Peucker algorithm. We would appreciate feedback in the form of comments on the performance of this algorithm, along with suggested criteria and test data for evaluating line simplification algorithms in general.

Acknowledgments

The authors are grateful to Dr. M. Visvalingam for commenting on the manuscript of this paper, and also Mr. S. M. Wise at SWURCC for providing information concerning the online retrieval of cartographic data.

References

Buttenfield, B.P.,1986, 'Digital Definitions of Scale dependent Line Structure', in M.Blakemore (ed) Proc. Auto Carto London (ICA, 1986), 497-506.

Carter, J.R.,1984, Computer Mapping: Progress in the '80's, (Association of American Geographers, Washington), Chapter 2.

Douglas, D.H. and Peucker, T.K.,1973, 'Algorithms for the reduction of the number of points required to represent a digitised line or its caricature', The Canadian Cartographer 10 (2) 112-122.

Jones, C.B. and Abraham, I.M.,1987, 'Line Generalisation in a Global Cartographic Database', Cartographica 24 (3) 32-45.

McMaster, R.B.,1987, 'The Geometric Properties of Numerical Generalization', Geographical Analysis 19 (4) 330-346.

Monmonier, M.S.,1986, 'Towards a Practicable Model of Cartographic Generalization', in M.Blakemore (ed) Proc. Auto Carto London (ICA, 1986) 257-266.

Robinson, A.H., Sale, R.D. and Morrison, J.L.,1978, Elements of Cartography, (Wiley, NY), Chapter 8.

Thapa, K.,1988, 'Automatic Line Generalization using Zero Crossings', Photogrammetric Engineering and Remote Sensing 54 (4) 511-517.

Waugh, T.C. and McCalden, J.,1983, Gimms Reference Manual. (Gimms Ltd., Edinburgh EH3 9E4), Edn 4.5.

White, E.R.,1983, 'Perceptual Evaluation of Line Generalization Algorithms', Unpublished Masters Thesis, University of Oklahoma.

Williams, R.,1987, 'Preserving the Area of Regions', Computer Graphics Forum 6 43-48.

Visvalingam, M and Kirby G.H.,1987, Directory of Research and Development based on Ordnance Survey Small Scales Digital Data, CISRG, University of Hull.

Appendix 1

This appendix describes the simplification of the line depicted in figure 4a. Figure 4b shows the method by which points are selected and figure 4c depicts the simplified line. The reader should refer to figure b in conjunction with the following text which describes the point selection procedure in detail.

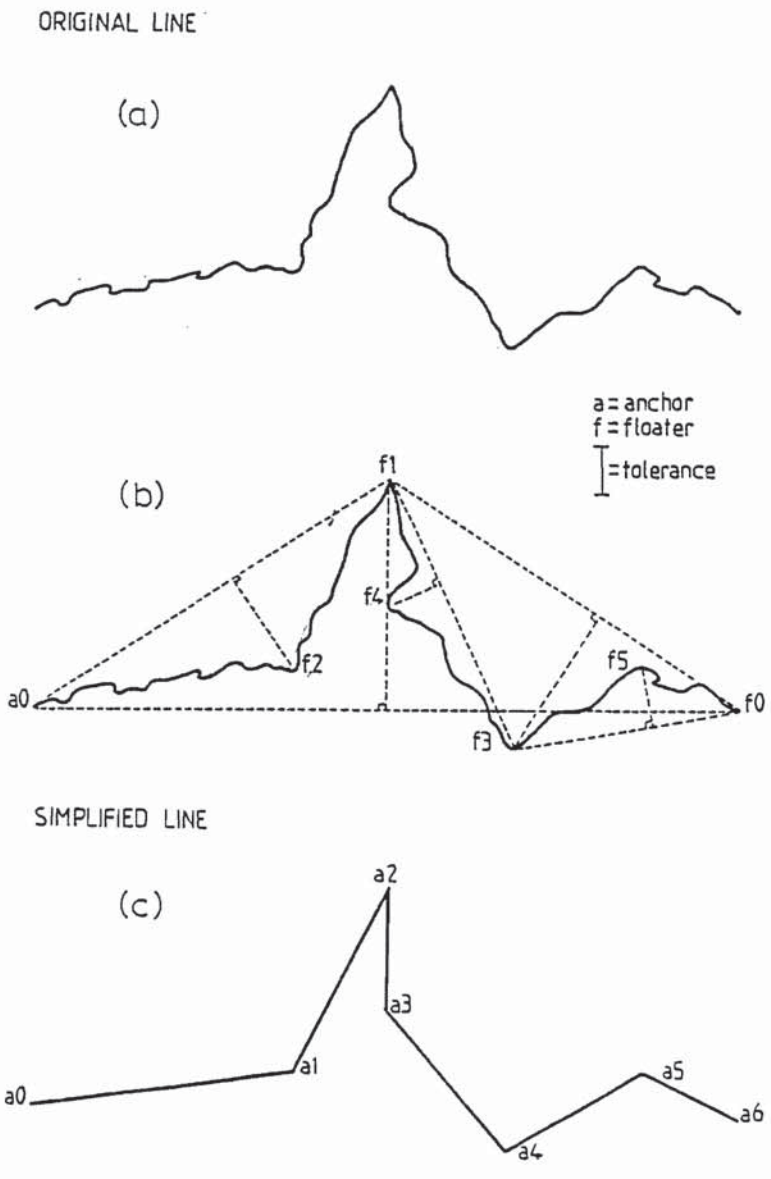

Figure 4.

1. Straight line segment a0 - f0 . Max offset at point f1.
2. Straight line segment a0 - f1 . Max offset at point f2.
3. Straight line segment a0 - f2 . All intervening points within tolerance.
   Anchor moves to point f2 (a1).
4. Straight line segment a1 - f1 . All intervening points within tolerance.
   Anchor moves to point f1 (a2).
5. Straight line segment a2 - f0 . Max offset at point f3.
6. Straight line segment a2 - f3 . Max offset at point f4.
7. Straight line segment a2 - f4 . All intervening points within tolerance.
   Anchor moves to point f4 (a3).
8. Straight line segment a3 - f3 . All intervening points within tolerance.
   Anchor moves to point f3 (a4).
9. Straight line segment a4 - f0 . Max offset at point f5 .
10. Straight line segment a4 - f5 . All intervening points within tolerance.
    Anchor moves to point f5 (a5).
11. Straight line segment a5 - f0 . All intervening points within tolerance.
    Anchor moves to point f0 (a6).
12. Simplification completed by joining the anchor points with straight line segments.

Appendix 2. Gimms program used to produce figure 2

```
*SYSPARM DIAGS=2
*FILEIN LINE
   FILEIN=10,FILEOUT=11
   TITLE='DRIVER FOR LINE PLOTS'
   LIMITS=470000,350000,558000,520000
   REDUCE=1000
*SAVE=11
*PLOTPARM PLOTTER
   *PLOTPROG
   *NEWMAP MAPSIZE=7.9,15.3 FRAME NOLOGO
   *GIMMSFILE=11
   *DRAWMAP
   *TEXT POSITION=5,14 SIZE=0.2 ALPHABET=15
   'REDUCE=1000'
*END
*STOP
```

Appendix 3. GIMMS program used to produce figures 3a-d.

```
*SYSPARM DIAGS=2
*FILEIN SEGMENT
   FILEIN=10 FILEOUT=11
   TITLE='HUMBERSIDE DISTRICTS SEGMENT FILE'
   LIMITS=4400,3800,5500,4800
   AUTONODE
   BEGIN
*POLYGON
   FILEIN=11
   FILEOUT=12
   NZONES=210
   MAXPTS=350
   ALPHA
   GENERAL=50,25,12
   EXCLUDE ZONE=OUT
*PLOTPARM PLOTTER
   *PLOTPROG
      *NEWMAP MAPSIZE=8,8 FRAME NOLOGO
      *GIMMSFILE=12
      *ORIGIN PLOT 3.5,3.5 MAP=4950,4300
      *DRAWMAP GENERAL=3
      *TEXT POSITION=0.5,7.5 SIZE=0.2 ALPHABET=15
      '*DRAWMAP GENERAL=3'
   *END
*STOP
```

Appendix 4. The Hull implementation of the Douglas and Peucker algorithm.

```
      PROGRAM PEUCKER

      INTEGER X(10000),Y(10000),GCODE(10000)
      INTEGER COUNTER,INCOUNT,NUMSEGS,NUMCOORDS,NFCO
      INTEGER XMIN,XMAX,YMIN,YMAX,DISTTOL

C     5 - READ FILE ; 6 - WRITE FILE ;

C     Read tolerance value from keyboard.
1     FORMAT ('PLEASE ENTER A TOLERANCE VALUE')
      WRITE (*,1)
      READ (*,*) DISTTOL

C     Read limits of data from file header.
2     FORMAT (I6,1X,I6,1X,I6,1X,I6)
      READ (5,2) XMIN,XMAX,YMIN,YMAX

C     Read the number of chains from file header.
      READ (5,*) NUMSEGS
C     Output the limits of this data.
3     FORMAT ('LIMITS=',I6,1X,I6,1X,I6,1X,I6,1X)
      WRITE (6,3) XMIN,XMAX,YMIN,YMAX
C     Output the number of chains.
4     FORMAT ('NUMBER OF SEGMENTS=',I4)
      WRITE (6,4) NUMSEGS

C     Main loop.
      DO 100,COUNTER=1,NUMSEGS
C     Read the number of coordinates in a chain from file header.
        READ (5,*) NUMCOORDS

C      Read in a chain of co-ordinates.
        CALL COORDSIN(NUMCOORDS,X,Y)
C     Generalize this chain using Douglas and Peucker method.
        CALL GENERALIZE(X,Y,NUMCOORDS,GCODE)

C     Count the number of filtered co-ordinates in this chain.
        NFCO=0
        DO 150,INCOUNT=1,NUMCOORDS
            IF (GCODE(INCOUNT).GT.DISTTOL) THEN
                NFCO=NFCO+1
            ENDIF
150     CONTINUE

C     Output the number of filtered co-ordinates.
5       FORMAT ('COORDINATE PAIRS WHICH EXCEED CHOSEN TOLERANCE=',I4)
        WRITE (6,5) NFCO
        WRITE (6,*)
C     Output the chain of filtered co-ordinates.
        DO 200,INCOUNT=1,NUMCOORDS
            IF (GCODE(INCOUNT).GT.DISTTOL) THEN
```

```
            WRITE (6,2) X(INCOUNT),Y(INCOUNT),GCODE(INCOUNT)
         ENDIF
200   CONTINUE
100   CONTINUE
      END

      SUBROUTINE COORDSIN(NOCDS,A,B)
      INTEGER NOCDS,A(NOCDS),B(NOCDS)
      READ (5,*) A,B
      RETURN
      END

      SUBROUTINE GENERALIZE(XCOORD,YCOORD,PTSINCHAIN,GENCODE)
      IMPLICIT NONE
      INTEGER PTSINCHAIN,I,J,CORDSTART,CORDEND,POINT
      INTEGER XCOORD(PTSINCHAIN),YCOORD(PTSINCHAIN)
      INTEGER GENCODE(PTSINCHAIN)
      REAL DIS,MAXDIS

C     Set the generalization code of the first and last points to 999999
C     and set the others to -1.
      GENCODE(1)=999999
      DO 30,I=2,PTSINCHAIN-1
         GENCODE(I)=-1
 30   CONTINUE
      GENCODE(PTSINCHAIN)=999999
      CORDSTART=1
C     Main loop to give a generalization code to each point.
      DO 50,I=1,PTSINCHAIN-2

C     Find the first point on the line to have a generalization code.
         DO WHILE (GENCODE(CORDSTART+1) .GT. -1)
            CORDSTART=CORDSTART+1
         END DO

C     Find the first point after this with a generalization code.
         CORDEND=CORDSTART+2
         DO WHILE(GENCODE(CORDEND) .EQ. -1)
            CORDEND=CORDEND+1
         END DO

C     Find the point with the maximum distance from the "Anchor-Floater"
      line
         MAXDIS=-1.0
         DO 40,J=CORDSTART+1,CORDEND-1
            CALL CLCDIS(DIS,XCOORD(CORDSTART),YCOORD(CORDSTART),
     -                  XCOORD(CORDEND),YCOORD(CORDEND),
     -                  XCOORD(J),YCOORD(J))
            IF (DIS .GT. MAXDIS) THEN
               POINT=J
               MAXDIS=DIS
            END IF
 40      CONTINUE
```

```
C     Give the point which is furthest away from the "Anchor-Floater"
C     line the generalization code of this distance away.
         GENCODE(POINT)=MIN(GENCODE(CORDSTART),
     -                      GENCODE(CORDEND),
     -                      NINT(SQRT(MAXDIS)))
50    CONTINUE
      END

      SUBROUTINE CLCDIS (DIS,XCDST,YCDST,XCDEND,YCDEND,XPT,YPT)

      IMPLICIT NONE
      INTEGER XCDST,YCDST,XCDEND,YCDEND,XPT,YPT
      REAL X1,Y1,X2,Y2,X3,Y3,LAMBDA,X,Y,DIS

      X1=XCDST
      Y1=YCDST
      X2=XCDEND
      Y2=YCDEND
      X3=XPT
      Y3=YPT

      IF (XCDST .EQ. XCDEND .AND. YCDST .EQ. YCDEND) THEN
         LAMBDA=0.0
      ELSE
         LAMBDA=(X1*(X1-X2-X3)+X2*X3+
     -           Y1*(Y1-Y2-Y3)+Y2*Y3)/
     -           ((X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1))
      END IF

C     Calculate the values of x and y, this is the co-ordinate of the
C     point where the perpendicular from the point in question joins
C     the line from the anchor to the floater.

      IF (LAMBDA .LT. 0.0) THEN
         X=X1
         Y=Y1
      ELSE IF (LAMBDA .GT. 1.0) THEN
         X=X2
         Y=Y2
      ELSE
         X=X1+LAMBDA*(X2-X1)
         Y=Y1+LAMBDA*(Y2-Y1)
      END IF

C     Calculate the square of the distance from (X3,Y3) to (X,Y)
      DIS=(X3-X)*(X3-X)+(Y3-Y)*(Y3-Y)

      RETURN
      END
```

Appendix 5. Data for Read's Island, Humberside

495249 497764 421588 422588

1

29

497483	497293	496832	496581	496150	495650	495660	495520
495519	495409	495249	495350	495530	495631	495711	495811
496052	496283	496724	496774	496914	497084	497454	497614
497714	497764	497754	497633	497483	422331	422341	422491
422500	422579	422588	422538	422508	422548	422558	422397
422277	422157	422047	422017	421927	421797	421707	421598
421638	421588	421609	421750	421830	421961	422182	422252
422332	422331