Topologically-Consistent Map Generalisation Procedures and Multi-Scale Spatial Databases[1]

P. M. van der Poorten, Sheng Zhou and Christopher B. Jones

Department of Computer Science,
Cardiff University
Newport Road, PO Box 916
Cardiff CF24 3XF
United Kingdom
{p.vanderpoorten, s.zhou, c.b.jones}@cs.cf.ac.uk

Abstract. An important requirement of multiscale spatial databases is that topological consistency is maintained both within individual features and between co-displayed features for all scales at which they may be retrieved. Here we show how a triangulation-based branch-pruning generalisation procedure can be enhanced to enable its output to be used to build topologically-consistent multiscale data structures. A major limitation of existing branch-pruning methods, of the lack of vertex filtering, is overcome by the application of a topologically consistent, vertex priority labelling procedure. The branch pruning generalisation method is also improved by the introduction of an edge re-sampling technique and the provision of control over single and double-sided application of pruning. Experimental results of the use of the techniques are presented.

1. Introduction

Visualisation and analysis of spatial data at multiple levels of detail is fundamental to many applications of geographical information. Traditionally the requirement for multiple levels of detail has been met by the production of topographic map series at a range of scales. This approach of maintaining discrete single-scale versions, or multiple representations, is currently reflected in geographical information systems and their associated spatial database technology. Thus primitive spatial objects in a GIS represent scale-specific map features using geometric primitives such as polylines and polygons. Some GIS do maintain multiple versions at a few different scales, but the database access facilities are typically limited to retrieval of the few stored, fixed-scale representations. Typically, the database is unable to access the geometry at intermediate levels of detail and as a consequence is unable to adapt effectively to the scale specific requirements of many GIS applications. There is a

[1] This work was supported by an ESRI Research Contract and by the UK EPSRC grant GR/49314

need therefore for multi-scale spatial databases that provide progressive access to multiple levels of detail of spatial objects.

1.1 Multi-scale spatial data access schemes

The idea of building data structures to support multi-scale access to simple spatial objects such as lines and polygons dates back to the early 1980s. The strip tree (Ballard 1981) organised the geometry of linear features hierarchically in a binary tree, the nodes of which represented approximating line segments by bounding rectangles (strips). The arc tree (Gunther 1988) provided a variation in which approximating line segments were indexed by ellipses. Another binary tree structure, the BLG-tree, was combined with R-tree indexing of spatial objects covering specified scale ranges within the Reactive data structure of van Oosterom (1994). Layered approaches, in which scale-prioritised geometry is allocated to pre-specified scale intervals, were introduced in the quadtree-based multi-scale line tree of Jones and Abraham (1986) and in the PR-file (Becker et al 1991).

Experimental implementation of these techniques has been based on basic line simplification procedures, typically the Douglas-Peucker algorithm. This has resulted in some major limitations of the resulting databases. One problem is that the hierarchical ordering of vertices generated by the algorithm may not correspond to a monotonic change in the threshold values of the respective vertices. Consequently some vertices low down in the hierarchy may provide more significant shape information than higher level vertices. Another more challenging problem is that the intermediate scale geometry that is retrieved cannot be guaranteed to be topologically consistent either with itself or with neighbouring map features.

The problem of topological consistency has been considered with regard to complete spatial objects (of which there may be multiple representations) by Bertolotto and Egenhofer (1999) and by van Putten and van Oosterom (1998) who presented schemes for maintaining consistency between these complete objects. Maintenance of topological consistency between the multiple realisations of individual polylines and polygons in a database requires line and polygon generalization algorithms that can be guaranteed to generate simplifications that are topologically consistent. This issue was considered in Jones et al (2000), who also presented a scheme for keeping track of different levels of topological consistency. Several algorithms have been presented for generalising lines in a topologically consistent manner, including de Berg (1998), Saalfield (1999), van der Poorten and Jones (1999) and Ai et al (2000), but there has been little progress in the application of such procedures for priority labelling of vertices in a multi-scale database in order to guarantee topological consistency across retrieved levels of detail of the line and area primitives.

In this paper we show how the topologically consistent, "branch-pruning" generalisation procedure of van der Poorten and Jones (1999) can be enhanced and its results used to improve the quality of multi-scale datasets used to construct a multi-scale database. "Branch Pruning" in essence involves identifying 'features' of a line (very roughly speaking, sections of the line between bends) and selectively removing them. The method is based on triangulating the space around the lines. This

procedure was chosen as the basis of multi-scale database construction because it provides more flexibility than any other topologically-consistent procedure with regard to controlling the style of generalisation and enabling simultaneous generalisation of multiple map features. The technique of Ai et al (2000) is very similar, but it operates only on a single map feature, pruning one side at a time. A fundamental limitation of the branch pruning technique in general is that, while simplifying shape, it does not systematically reduce the number of vertices. A method is required therefore to filter redundant vertices while maintaining topological consistency. The previous implementations of branch pruning also suffer from other limitations with regard to the introduction of discontinuities (or "stumps") following branch deletion and the lack of explicit control over whether branches refer to one or other, or both, sides of the line.

In the remainder of the paper we provide in Section 2 an overview of the aspects of a multi-scale spatial database architecture that provides support for multi-resolution representation of individual polylines and polygons. This is followed in Section 3 by the description of triangulation-based branch pruning procedures which builds upon the ideas in van der Poorten and Jones, but extends them to apply to polygons, in addition to polylines, to incorporate a solution to the problem of introducing stumps, and to provide an option for single or double-sided application of the branch pruning procedure. Examples of its application to real data are provided. Section 4 describes and illustrates the results of a triangulation-based topologically consistent priority-labelling procedure that can be used to post-process the results of branch pruning of multiple map features in order to remove the potentially large number of redundant vertices which would otherwise be present. The method implemented here adopts the principles of the Douglas-Peucker algorithm and is used here simply for point reduction, rather than shape simplification. Experimental results of applying the procedure to branch-pruned data are given for various combinations of branch prune metric and filter tolerance values. The paper concludes in Section 5 with a summary of the results and of future work.

2. Multi-Scale Spatial Access Schemes for Polylines and Polygons

Here we give an overview of the geometric data storage characteristics of a multi-scale spatial database, based on Zhou and Jones (2001a), that can provide access to individual spatial objects at multiple levels of detail. We are concerned here only with the issue of storage of the scale-priority attributed geometry and not with other issues such as selection of whole map features, or the maintenance and update of the database objects. The conceptual model of the multiscale spatial database represents map features as Multi-scale Spatial Objects, or MSOs, which have an application specific class, cover a resolution range R_{MSO} and reference one or more Multiscale Geometry Objects (MGEOs). The MGEOs have a geometry type t_{MGEO}, a resolution range r_{MSO} and, in the case of simple polylines and polygons which we consider here, an ordered set of vertices V_{MGEO}. A vertex consists of the components (*MGEOID*, *vid*, *R*, *vsn*, *x*, *y*), where *MGEOID* is the identifier of the parent MGEO, *vid* is the identifier of the vertex, *R* is the set of resolution ranges of the vertex, and *x* and *y* are

the geometric coordinates in 2D space. The implementation need not store all of these items explicitly for every vertex.

The term resolution refers here to a numerical value that can be used to determine the applicability of the vertex to a specified query scale. In practice it is equivalent to, or a function of, a tolerance value that has been used to control the degree of generalisation of the respective geometric object. In the case of the Douglas-Peucker (1973) algorithm, its tolerance value, when divided by a map scale denominator, may be regarded as a measure of the smallest discernable feature on the map.

The significance of a resolution value attached to a vertex varies according to the nature of the generalisation algorithm that was used to produce it. We can distinguish primarily between sub-setting and non-sub-setting procedures. In the former, each vertex in the representation is present at the most detailed level of representation, while in the latter new vertices may be introduced in the course of generalisation. We can also distinguish between continuous and non-continuous vertices. A continuous vertex is one that appears across a single range of resolutions of the geometry. A non-continuous vertex may appear in separate discrete ranges of resolution. The Douglas-Peucker algorithm results in continuous sub-setting vertices in that each vertex represents a range of resolution from the most detailed to some intermediate or extreme level of simplification. The original branch pruning procedures result in continuous sub-setting vertices, but the modification of branch pruning to avoid stumps, that we present in this paper, results in non-sub-setting vertices in that new vertices can be introduced (and sometimes subsequently be removed) following the elimination of a branch.

2.1 Implementation schemes

There are many possible ways of implementing a spatial database to support access to MGEOs, following on from the techniques referred to in Section 1.1. Zhou and Jones (2001a) have demonstrated the practicality of a layered scheme, similar to that of Becker et al (1991), in which vertices of an MGEO are grouped initially with respect to a scale interval partition and then within a layer with regard to space, using R-trees to index sub-sequences of vertices. Resolution values of vertices within a layer may be mapped to a single value that is representative of the entire layer, which can be adequate in the case of continuous sub-setting vertices. Alternatively, for example in the case of continuous, non-subsetting vertices, a pair of values representing the range of the vertex may be stored.

An alternative to layered schemes is a tree scheme that may include internal spatial indexing, or simply complete object indexing in the manner of the BLG-tree.

3 Feature-Based Line and Polygon Generalisation with Branch Pruning

3.1 Overview of triangulation-based branch pruning

The technique of branch pruning is based on the principle of eliminating discernable features of a line according to shape criteria. A feature corresponds to one or more bends in the line that introduce protuberances or embayments, and it may be hierarchical in the sense of having sub-features at multiple levels of detail. In the triangulation method, the set of lines and polygons to be generalised is triangulated with a constrained Delaunay triangulation (CDT). The paths of sequences of neighbouring triangles are then used to approximate the location of branches of the skeleton of the lines and polygons (Ferley et al 1997; Gold 2000). The "true" skeleton, or medial axis transformation, represents the locus of points that are equidistant from the boundaries. It's branches and sub-branches may be equated with features of the lines and polygons (Lee 1982) at their respective levels of detail. Within the CDT, the sets of triangles that represent features can be used to calculate metrics that may be used to distinguish between different shaped features. The metrics are based on the dimensions of the constituent triangles and the set of constraining edges that constitute the corresponding feature.

3.2 Triangulation components

We define here some concepts to be used in the analysis of the CDT. Edges of the triangulation that belong to an original line (and hence constrain the triangulation) are described as *real*, while those that belong to the bounding box *external* and all others are *virtual*. Two triangles that have an edge in common are described as *internal neighbours* if the edge is *virtual* and *external neighbours* if it is *real*. Triangles that are internal neighbours are said to be *connected*.

Triangles are divided into three basic types. A triangle with two real edges is termed a *leaf* triangle, while one with one such edge is termed a *trunk* triangle and one with none is a *branching* triangle. Figure 1 illustrates this categorisation.

A *branch* in the CDT of an open line is a contiguous set of connected triangles, bounded by a sequence of real edges belonging to the line, plus a single virtual edge, referred to as the base edge of the branch (see Figure 2). The sequence of real edges is defined to be the *feature* of the line that the branch represents, ideally it should coincide with the visual feature referred to above. The two vertices of the branch's base edge are the first and last vertices of the feature.

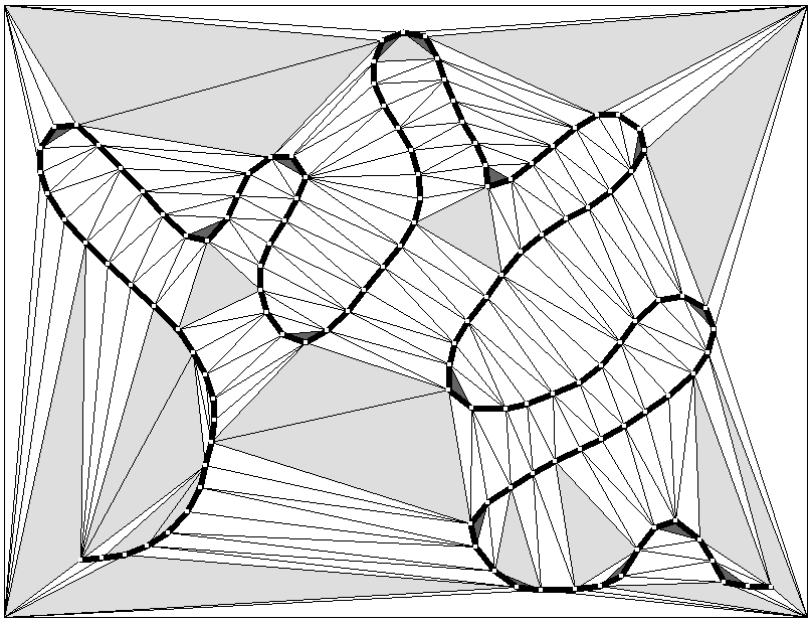


Fig. 1. A line and its corresponding constrained Delaunay triangulation. Leaf triangles are dark grey, trunk triangles are white and branching triangles are light grey.

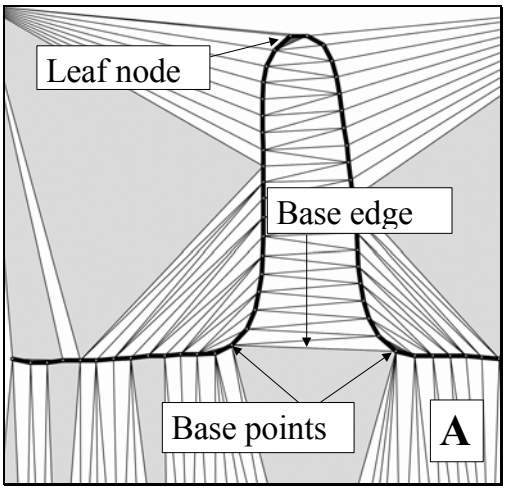


Fig. 2. A feature, a branch, and its base edge

A *path* is an ordered sequence of connected triangles. Paths cannot backtrack, that is, they may not cross the same virtual edge more than once, but may form a loop.

We divide each triangle type (branching, trunk, leaf) into several subtypes, based on the role they play in the triangulation structure. This is mostly for purposes of

computational efficiency only and most of these details are not covered here. However, one important sub-type is the *root triangle* (for the single line case equivalent to *type I* triangles in Ai et al 2000). Such a triangle forms the 'root' of an entire feature. One way to define such a triangle is with regard to a *leafward edge*. A *leafward edge* is an edge of triangle, say T, which satisfies the following criteria:

- it is virtual
- it lies between two vertices of the same line of the original dataset
- all triangles which can be reached from it by means of a legal path (i.e. without passing through T) have all their vertices on that same line.

A *root triangle* is a branching triangle with exactly one such edge. Note that some branching triangles will have two such edges, while others will have none.

We can now define the *rootward edge* of triangle T as being that which (a) is *virtual* (b) lies between two vertices of the same line of the original dataset and (c) from which a path can be found to reach the *root triangle* of that branch without passing through T. See Figure 3.

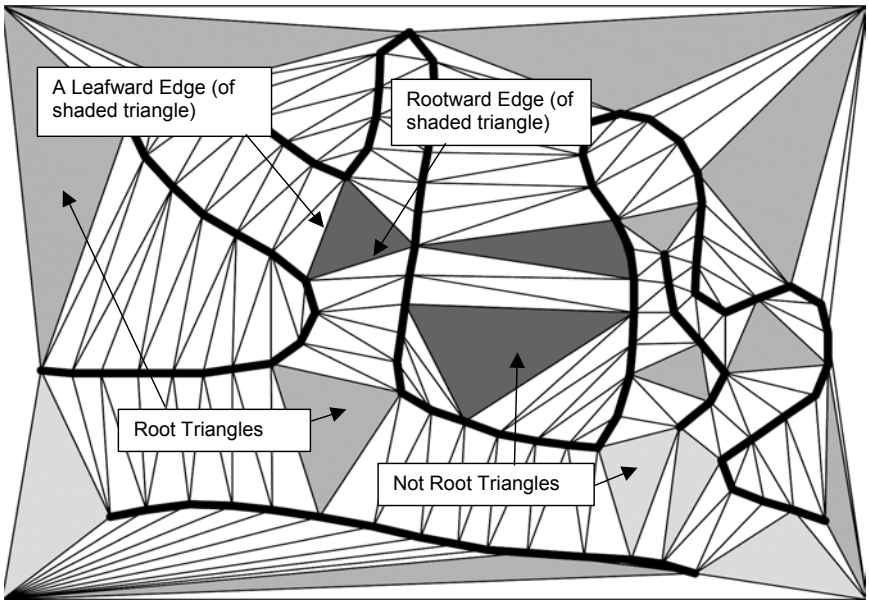


Fig. 3. Root triangles (shaded medium grey)

A further important distinction is between *interior* and *exterior* triangles. This arises in the presence of polygons within the original line set. *An interior triangle* is one all of whose vertices lie on a single line and from which no path can be traced to connect it to a triangle with any vertex on a different line. That is, it is a triangle lying within a polygon. Any triangle not *interior* is deemed *exterior*. See Figure 4.

The principle significance of this, is that within a polygon the 'root'/ 'non root' distinction does not apply. *All* interior branching triangles are roots to three branches.

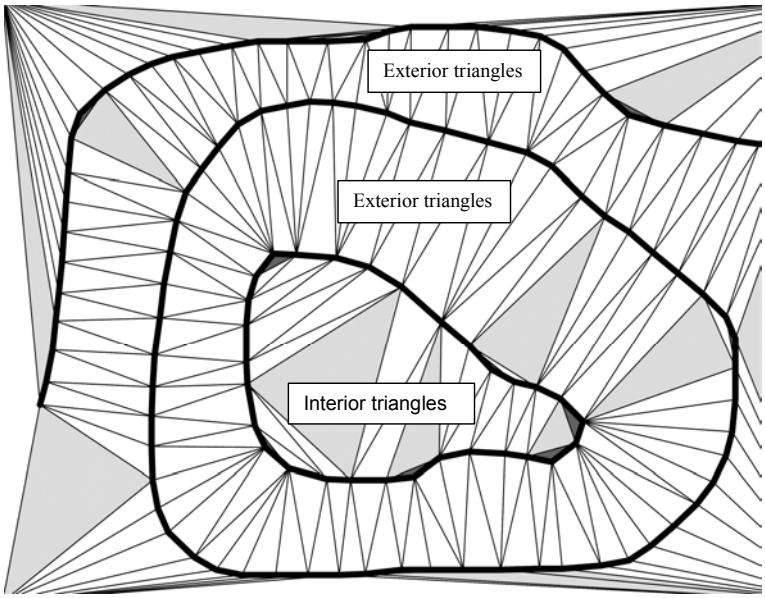


Fig. 4. Interior *vs*. exterior triangles

3.3 Branch statistics

Analysis of the triangulation as above reveals an implicit hierarchy of features. Features stem from the leafward edges of the root nodes, continuing in a leafward direction from triangle to (connected) triangle, while sub-features stem from the leafward edges of branching triangles. We can now calculate various statistical properties (‘*metrics’*) relating to each branch and sub-branch, in order to allow decisions to be made about which features to remove. By basing such decisions on different *metrics* we can achieve different *styles* of generalisation.

A dozen such statistics have been devised. Only two of these metrics are used in this paper (path length, average width). These, plus those necessary for their derivation, are described below. The *true error* metric is listed as the availablity of such a metric is clearly important for any line generalisation procedure. Other metrics (e.g. *boundary length)* are not discussed in this paper.

- The *area* of the branch is the total area of all its component triangles.
- To define the *length* of a branch we define the *node length.* For a trunk triangle this is the distance between the midpoints of its two internal edges. For a leaf triangle it is that from the midpoint of its (single) internal edge to its opposing vertex. A branching triangle has two *node lengths* depending on which sub-branch one is measuring. The length of the branch is obtained by summing the node lengths of all the triangles that form the branch. The length of a complex

branch is considered to be the length of its longest path – we follow the branch from its baseline, taking the longest branch at each junction.

- *Branch height* is calculated by summing *node heights* of a branch's constituent triangles. *Node height* is defined to be half the height of the relevant triangle (taking the rootward edge as the base).
- *Average width* is defined to be a branch's total area divided by its height.
- *True error* of a branch is the displacement error that would be introduced into the generalisation if the relevant branch were to be deleted.

3.4 Details of the method

Initially, the smallest branch of the triangulation, according to the selected metric, is identified. The segment of the line that defines this branch is removed and replaced by its baseline, and the affected area of the triangulation is updated. This process is repeated until the relevant metric value of the smallest remaining branch is above the given threshold value. It is also possible to specify thresholds for a combination of different metrics and delete all branches that fall below *all* of the relevant thresholds. The *true error* metric may be combined with others to ensure control over locational accuracy.

Figures 5 and 6 illustrate the application of branch pruning with the single metrics of *average width* and *length* respectively. Of particular note are the effects on the coastline, the rivers on the right and left hand sides of the map and the roughly U-shaped object adjacent to the coast-line. When pruning by width the narrower left hand river is removed completely by figure 5B, the wider right hand one by figure 5C. When pruning by length both rivers are retained, but lose their shorter tributaries.

It should be remarked that the dataset illustrated here consists of contours, for which no selection operator has been applied to provide an appropriate contour interval for the map scale. The dataset has been chosen as it constitutes a challenging test of the maintenance of topological consistency among multiple, often densely spaced features. It is not intended and does not serve as a demonstration of appropriate terrain model generalisation. Note also that the original dataset includes the un-closed contours that are apparent in the figures.

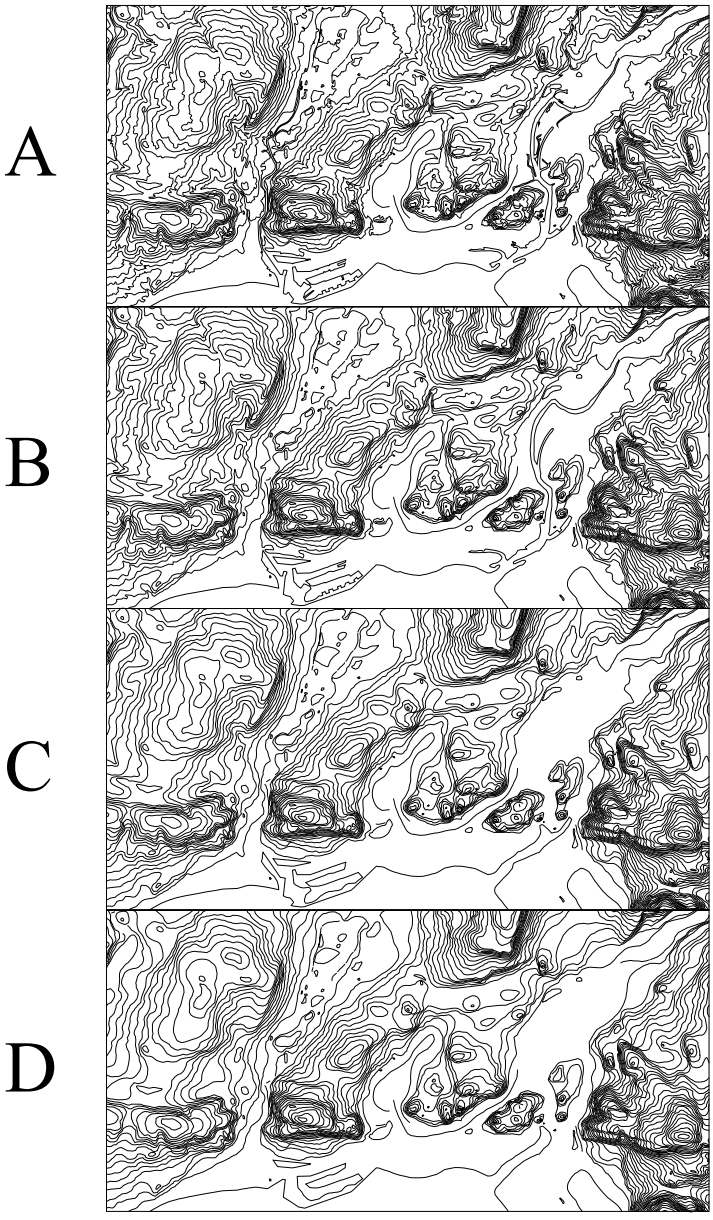


Fig. 5. Branch pruning – *A* original, width metric in metres *B* 64, *C* 128, *D* 256 (map size: 15 by 7.5km. Source data: ©Ordnance Survey® Crown copyright 2001)

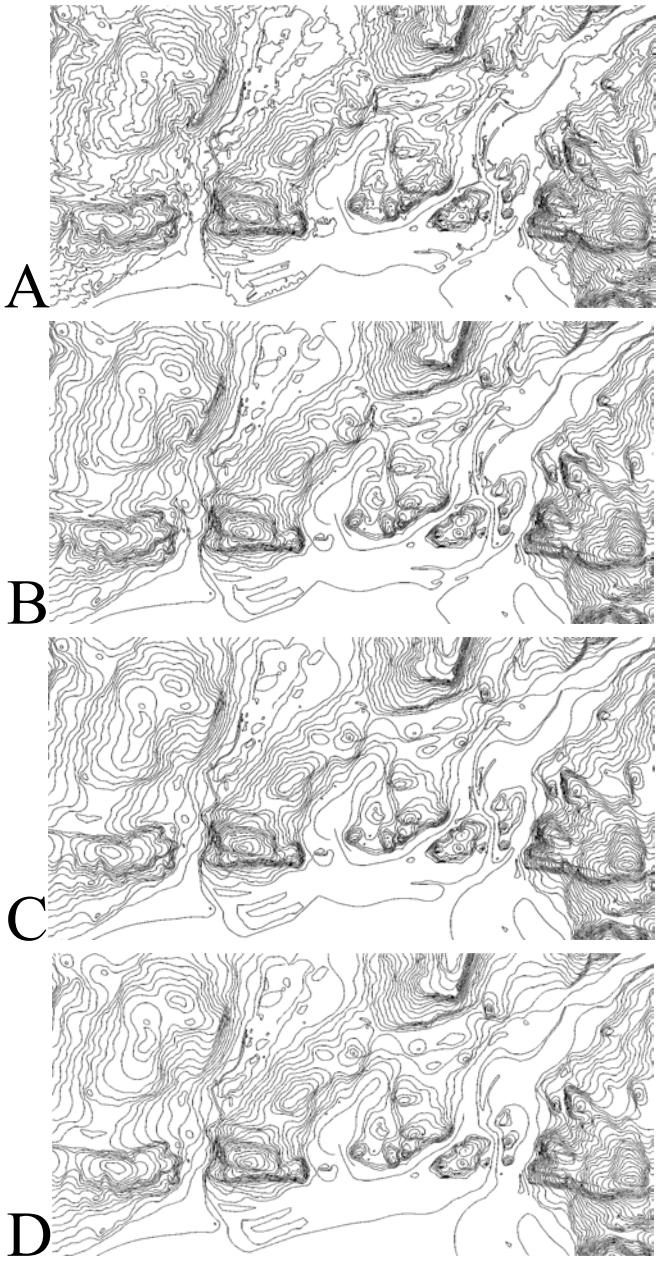


Fig. 6. Branch pruning – *A* original, length metric in metres *B* 128, *C* 256, *D* 512 (map size: 15 by 7.5km Source data: ©Ordnance Survey® Crown copyright 2001)

3.4 Two-sided or single-sided dynamic triangulation

Because this procedure makes use of dynamic re-triangulation, only updating the triangulation in the region affected by the deletion of a branch, it can deal with (multiple) lines in a completely two-sided fashion. However the procedure also allows the option of single sided pruning, with the ability to specify which side each line in the dataset is to be pruned from on a line-by-line basis. This could be particularly useful when considering features such as coastlines in which promontories such as peninsulas only exist as areal features on the landward side.

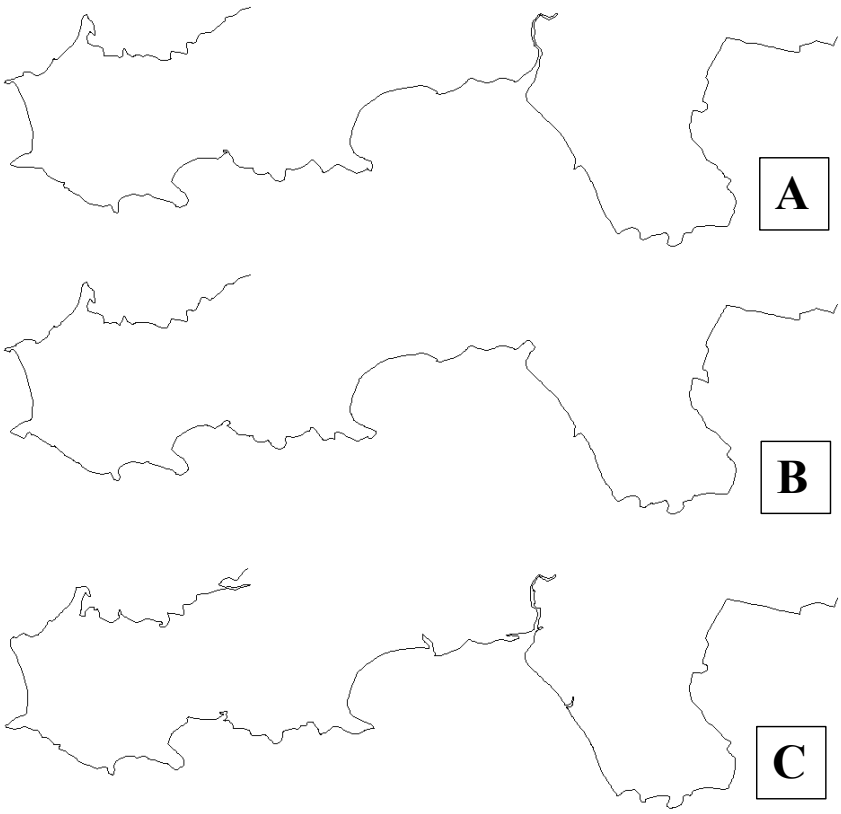


Fig. 7. Single (B and C) and dual (A) sided pruning

Figure 7B shows a (coast)line generalised (using the *width metric*) from one (seaward) side, figure 7C shows the same line generalised only from the reverse (landward) side, while figure 7A shows generalisation from both sides simultaneously. Note that certain features are retained when generalised from both sides yet removed when pruning is restricted to one side. This is because when

pruning is allowed on both sides, those features are slightly enlarged by the removal of their indentations from the reverse side.

3.6 Resampling

Resampling is the practice of adding co-linear vertices to the straight line left by the deletion of a branch. This helps greatly to smooth the resulting generalisation and avoids drastic changes in the level of detail from one part of the line to another. It also prevents the creation of what we call *stumps*. Figure 8A shows a line with a feature eligible for pruning. In this case the removal would leave an inappropriate stump because the base of the feature has not quite been correctly identified (in many cases a perfect cut may not be possible due to the configuration of original vertices). This is seen in figure 8B.

The underlying problem is that making such cuts creates line segments that may have a large vertex spacing compared to the distance between the line and a neighbouring line. The immediate effect of resampling following the removal of the branch is shown in Figure 8C. The new vertices lead to the creation of new smaller branches, the pruning of which smoothes the stump left by the pruning of the original branch, giving the final result shown Figure 8D. Further smoothing could result from resampling of cuts created by the subsequent removal of the new branches. In general resampling results in smoother generalisations, at the expense of increased processing time.

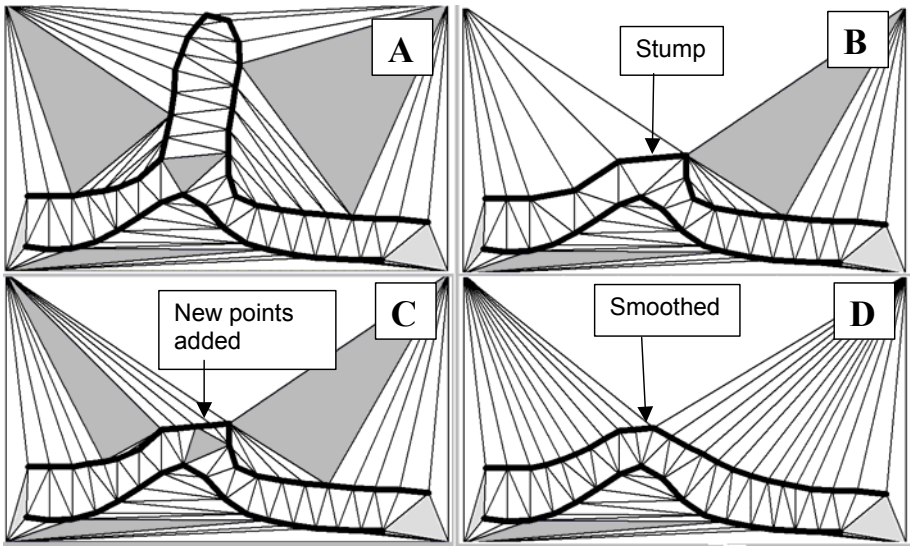


Fig. 8. Resampling.

4. Triangulation-based Topologically-consistent vertex priority labelling (TCL)

Here we describe a triangulation-based procedure for attaching resolution, or scale priority, values to the vertices of multiple map features in a manner that ensures topological consistency. The procedure operates on geometry in which the vertices have been given an initial priority value, using a generalisation procedure that need not be guaranteed topologically consistent. In our current implementation we use the Douglas-Peucker criterion (distance to a base-line) for the initial stage of priority labelling. This may be regarded as analogous to the BLG-tree of van Oosterom (1994). Compared with the BLG tree, we made the following improvements in the initial priority labelling procedure (Zhou and Jones 2001b):

- Priority promotion: when a vertex has a priority value P_c larger than that (P_p) of its parent vertex in the BLG-tree, the priority value of the parent vertex will be promoted to P_c so that priority values will always decrease monotonically on any path down from the root and the correct order of retrieval can be maintained.
- Feature sectioning based on convex hull: as the Douglas-Peucker criterion is used to calculate priority value, it is very important to select the proper vertices as the starting point of the process. For closed polylines, we use the two end points of the longest diagonal of its convex hull to divide the polyline into two sections and them process them separately; for open polylines, we apply the above convex-hull based method recursively to divide a polyline into one or several sections and then process them separately. This method solves the so-called problem of "extending beyond endpoints" (Gunther 1988, pp96).

The procedure does not guarantee that the retrieved result will be topologically consistent.

4.1 Scale priority dimension scan, topological inconsistency detection and removal

To detect and remove topological inconsistency within the whole scale range of the dataset, we designed an algorithm which starts from the smallest priority value (i.e. finest resolution, and hence largest scale) and scans the entire scale priority dimension until the largest scale priority value is reached, detecting and removing any topological inconsistency encountered. In this way, all potential query scale values falling into the scale range of the dataset will have been examined. We regard this method as a case of "progressive generalisation" (Zhou and Jones 2001b). Here is the outline of the algorithm:

- Step 1: Build an index ***I*** for all vertices in the dataset, sorted primarily by their priority values, and an empty list ***L*** that will be used to store vertices whose removal will cause inconsistency;
- Step 2: While ***I*** is not empty, remove v_i, the vertex with smallest priority value p_0, from ***I*** and test if the insertion of line segment v_{i-1}-v_{i+1} (the two adjacent vertices of v_i in the feature to which v_i belongs) will cause any topological inconsistency (intersection with other segments on the same feature or on other features);

- Step 2a: If no inconsistency occurs, remove v_i from the feature that owns it, and insert segment v_{i-1}-v_{i+1}. If ***L*** is not empty, raise the priority values of all vertices in ***L*** to p_0, and reinsert these vertices into ***I*** and empty ***L***. Go back to step 2.
- Step 2b: If inconsistency occurs, insert v_i into ***L*** and go back to step 2.

For a dataset with N vertices, this algorithm requires $O(N)$ time in the best case while each step a vertex can be labelled and removed. In the worst case while no vertex can be removed until the extent of the dataset is reached, it runs at $O(N^2)$ although this is unlikely for real datasets.

4.2 Implementation and experiment

The above algorithm has been implemented using C++. A dynamic constrained Delaunay triangulation procedure is used for inconsistency detection. An initial triangulation is computed for the whole dataset. Subsequently when a vertex is removed from the triangulation or a new constrained line segment is inserted into the triangulation, the triangulation is updated locally, which on average can be done in constant time. The process of consistency checking requires testing the potential new edge v_{i-1}-v_{i+1} for intersection with existing constraining edges in the triangulation. This is done by stepping through the triangles in the region of the triangulation between v_{i-1} and v_{i+1} checking for the presence of constraining edges along the path of the new edge. Note that in the event of a non-constraining edge coinciding with the path of the new edge, there is no need for further checks.

The test dataset is the contour layer of an Ordnance Survey sample Land-Form® tile (grid ref. ss68) at 1:10,000, which contains 846 polylines and 85024 vertices. The program was run on a notebook PC with mobile PIII 850MHz CPU and 128MB RAM.

4.3 The issue of proximity inconsistency

The TCL method presented here can be easily extended to PCL (P for proximity) for handling proximity inconsistency (vertex or line segment are too close to each other). For PCL, the intersection search procedure in TCL (see 4.1) will be replaced by a proximity search procedure such as (Jones and Ware 1998).

5. Combining TCL with branch pruning

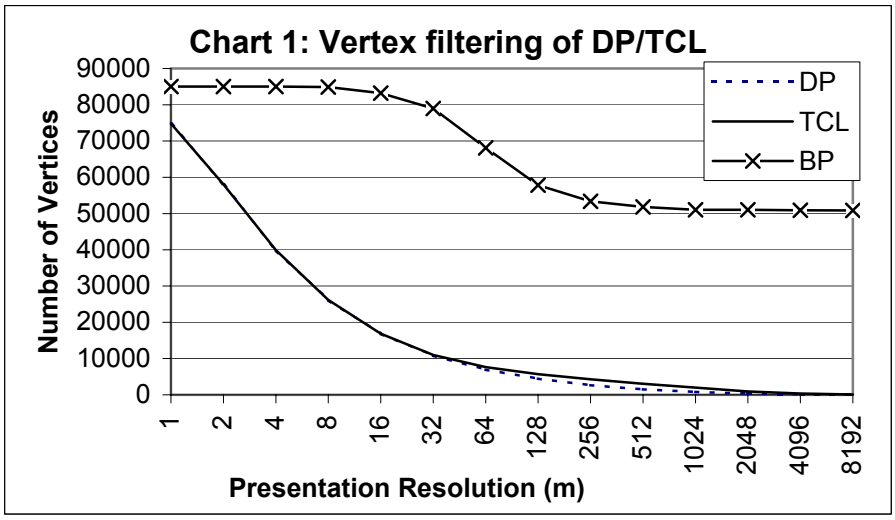
The simple strategy for inconsistency removal in the previous section is not very satisfactory due to the fact that it can remove at most one vertex each time. Therefore, some singular vertices may be retained at Douglas-Peucker threshold values much larger than those at which such vertices would normally be deleted, due to the close proximity of other features making the removal of such vertices impossible without the introduction of topological inconsistencies. Furthermore it is subject to other limitations of the Douglas Peucker algorithm for purposes of line generalization

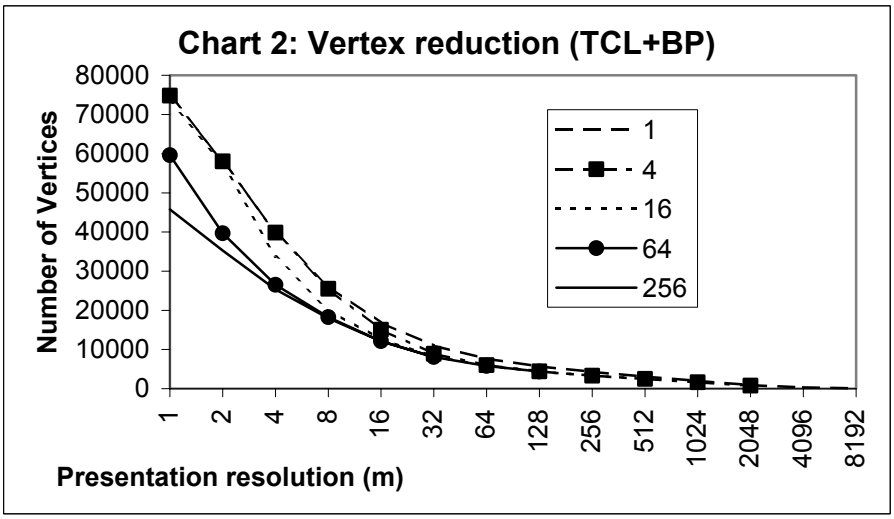
(Visvalingam and Whyatt 1991). In order to improve the result, we have combined the results from branch pruning with the TCL method, so that the Douglas-Peucker algorithm is functioning simply as a filtering procedure which removes redundant vertices from the branch pruning results, enabling the vertex density of branch pruning to be adapted to the display resolution. In Table 1 we present the results of applying the topologically-consistent filtering procedure to the branch pruned dataset for several combinations of branch prune metric and Douglas-Peucker tolerance. The table illustrates the failure of branch pruning by itself to filter vertices and the major effect of our post-processing operation in reducing numbers of vertices.

We have implemented a labelling procedure, whereby all branch prune labelled vertices of a dataset are relabelled with Douglas-Peucker style tolerances to facilitate mapping to required display resolutions. This is based on choosing a ratio of the branch prune metric to filter tolerance. Ratios that we have used in practice for branch metrics such as average width and length are typically in the range 4 to 8. Data derived from the labelling procedure have been used to create a layered multi-scale spatial database that is accessed by a Java servlet to create an experimental web demonstrator.

The initial constrained Delaunay triangulation time ranged between 2.4 and 4.0 seconds for numbers of vertices in branch pruned datasets ranging between 50,862 and 85,024 respectively. For the same range of numbers of vertices, the processing time to provide a topologically consistent labeling varied between 87.7 and 155 seconds, based on averaging 10 runs for each dataset. For the same range of datasets, the numbers of failed attempts at deleting vertices (i.e. ones that resulted in topological inconsistencies) ranged from 3,926,863 to 6,992,492. It should be stressed that the operation of vertex labeling is one that should be carried out at the time of building a database.

Chart 1 demonstrates the vertex filtering effects of DP and TCL in comparison to BP(using the same DP tolerance for branching pruning). Chart 2 shows the result of combining TCL and BP (using various larger brunch pruning tolerance values).





Figures 9 illustrates the TCL method applied on the original dataset, retrieved with three resolution values of 12.5m, 25m and 50m and plotted at scales of 1:125,000, 1:250,000 and 1:500,000 respectively.

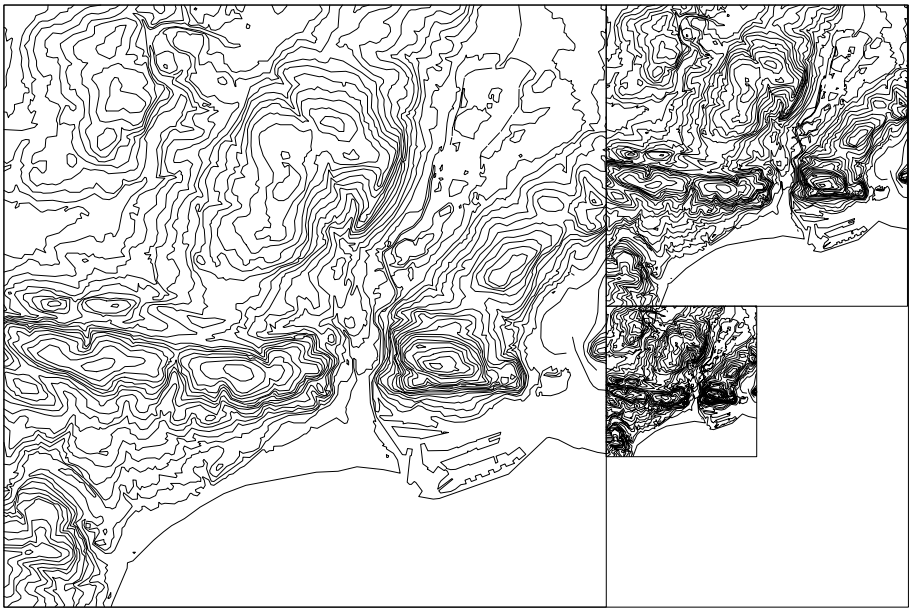


Fig. 9. TCL on the original dataset, plotted at 1:125,000, 1:250,000 and 1:500,000 (Source data: ©Ordnance Survey® Crown copyright 2001)

Figure 10 illustrates the application of the method to branch-pruned data that have been displayed with the same parameter values as those of Figure 9.

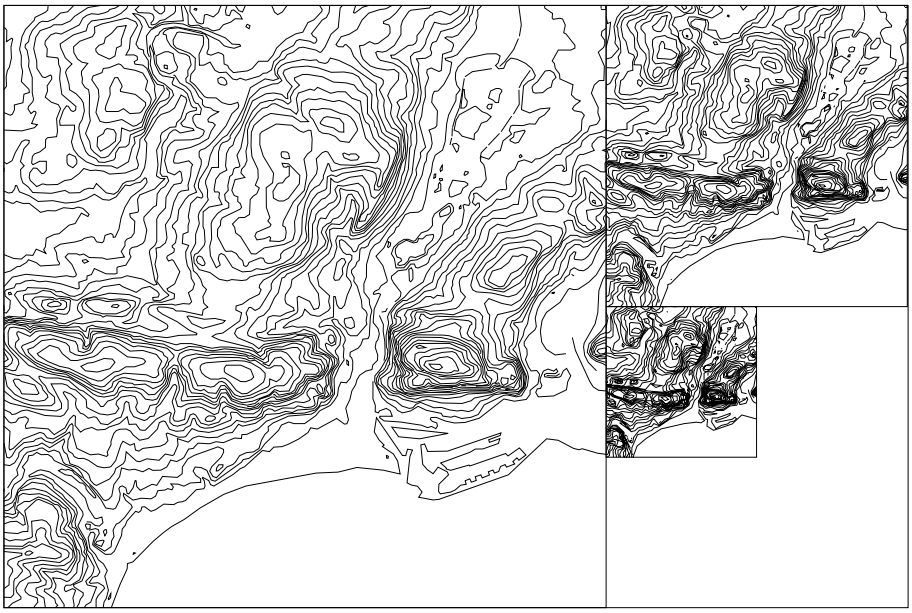


Fig. 10. TCL with Branch Pruning (ratio of BP to DP tolerance is ~5)(Source data: ©Ordnance Survey® Crown copyright 2001)

Figure 11 illustrates the use of the Douglas-Peucker algorithm only (using a tolerance of 12.5m), and highlights an example of the many topological inconsistencies that result.

5 Conclusions and Discussion

In this paper we have introduced several important enhancements to the branch pruning method of line generalisation to facilitate the use of the procedure for pre-processing geometry that may be stored in a multi-scale spatial database. The method is of particular interest in that it provides topological consistency for all levels of detail of ensembles of line and polygon features. A topologically consistent filtering procedure has been implemented for purposes of vertex priority labelling, to enable the vertex density of branch-pruned features to be adapted to the resolution of the map display. We have also implemented a re-sampling procedure to avoid the introduction of stumps at the base of pruned branches and we have provided explicit control over the use of single and double sided pruning procedures.

The approach described has been implemented in the context of a multi-scale database that is linked to a web server, using branch-pruned and filtered geometry for a single pruning metric. It is possible to envisage that future multi-scale databases could be implemented to exploit the versatility of branch pruning, by storing multiple

branch metrics that enable the style of generalisation to be modified online. The existing implementation is also limited, for example with regard to the small number of map generalisation operators that are supported. Current research is investigating the integration of the branch-pruning generalisation methods with conflict resolution procedures to ensure adequate separation of plotted map features through the application of selection, displacement and amalgamation. A further topic of future research is the development of incremental update procedures to maintain large databases of topologically consistent multi-scale data.

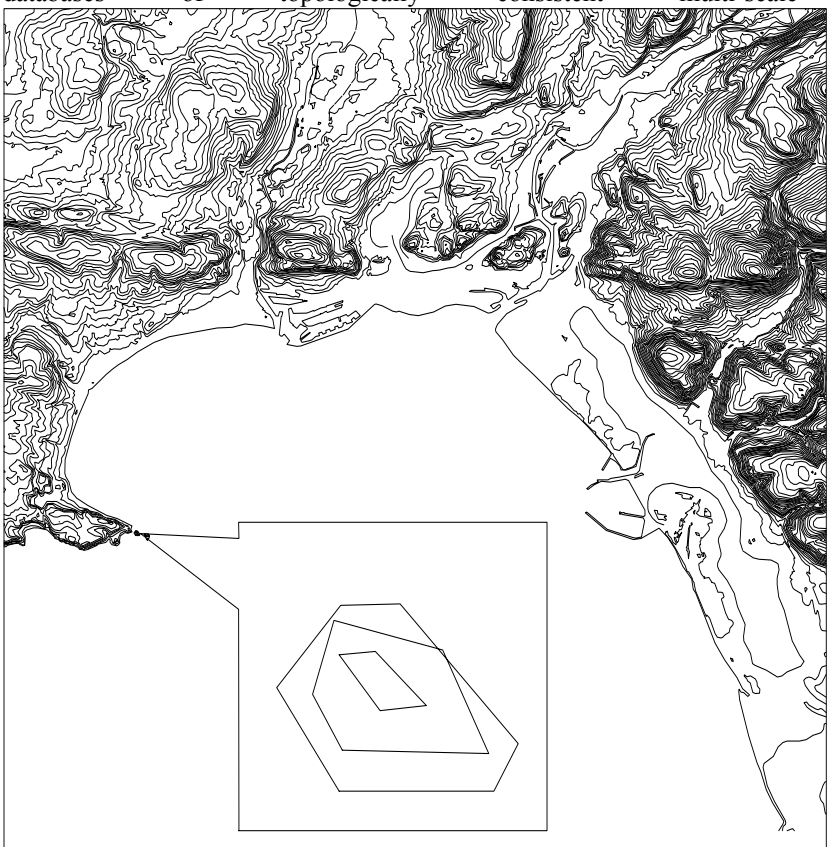


Fig. 11. Douglas-Peucker only - conflicts occur (DP tolerance = 12.5m) (Source data: ©Ordnance Survey® Crown copyright 2001).

References

Ai, Tingua, Guo, RenZhong, Guo and Yaolin, Liu, August 2000. "A Binary Tree Representation of Curve Hierarchical Structure Based On Gestalt Principles", *Proceedings 9th International Symposium on Spatial Data Handling*, sec. 2a, 30-43.

Ballard, D. 1981. "Strip trees, a hierarchical representation for curves." *Communications of the ACM* 24, 310-321.

Becker, B., H.-W. Six, et al. 1991. "Spatial priority search: an access technique for scaleless maps". *ACM SIGMOD* 20(2), 128-137.
Bertolotto, M. and M. J. Egenhofer 1999. "Progressive vector transmission". *7th ACM Symposium on Advances in Geographic Information Systems*, Kansas City, MO, ACM Press, 152-157.
De Berg, M., M. van Kreveld, et al. 1998. "Topologically correct subdivision simplification using the bandwidth criterion." *Cartography and Geographic Information Systems* 25(4), 243-257.
Douglas, D. H. and T. K. Peucker 1973. "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature." *The Canadian Cartographer* 10(2), 112-122.
Ferley E, Cani-Gascuel M-P and Attali D., 1997. "Skeletal Reconstruction of Branching Shapes", *Computer Graphics Forum*, 16(5), 283-293.
Gold, C. 2000. "Primal/Dual Spatial Relationships and Applications", *Proceedings 9th International Symposium on Spatial Data Handling*, sec. 4a, 15-27.
Günther, Oliver 1988. *Efficient structures for geometric data management*. Springer-Verlag.
Jones, C.B. Abdelmoty A.I, Lonergan, M.E.,van der Poorten, P.M., and Zhou, S. 2000. "Multi-scale spatial database design for online generalisation". *Proceedings 9th International Symposium on Spatial Data Handling*, sec. 7b, 34-44.
Jones, C. B. and I. M. Abraham 1986. "Design considerations for a scale-independent database". *Second International Symposium on Spatial Data Handling*, Seattle, International Geographical Union, 384-398.
Jones, C.B., Bundy, G. L. and Ware, J.M., 1995. Map Generalisation with a Triangulated Data Structure. *Cartography and Geographical Information Systems,* 22(4), 317-313.
Jones, C.B. and J. M. Ware, 1998. Proximity Search with a Triangulated Spatial Model. *The Computer Journal,* 41(2), 71-83
Lee, D.T, July 1982. Medial Axis Transformation of a Planar Shape, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 4, 363-369.
Saalfield, A. 1999. "Topologically consistent line simplification with the Douglas-Peucker algorithm." *Cartography and Geographic Information Science* 26(1), 7-18.
van der Poorten, P.M, Jones C.B., August 1999. "Customisable Line Generalisation using Delaunay Triangulation", CD ROM proceedings of the *19th ICA conference* Ottawa, section 8.
van Oosterom, P. 1994. *Reactive Data Structures for Geographic Information Systems*. Oxford, Oxford University Press.
van Putten, J. and P. van Oosterom 1998. "New results with generalised area partitionings". *8th International Symposium on Spatial Data Handling*, Vancouver, International Geographical Union.
Visvalingam M and J.D. Whyatt 1991. "Cartographic algorithms: Problems of implementation and evaluation and the impact of digitising errors". *Computer Graphics Forum* 10(3), 225-235.
Zhou, S. and Jones, C.B. 2001a. "Design and Implementation of Multi-scale Databases", Advances in Spatial and Temporal Databases, 7th International Symposium, SSTD 2001, Proceedings. Lecture Notes in Computer Science 2121 Springer 2001, 365-386
Zhou, S and Jones, C.B. 2001b. "Multi-Scale Spatial Database and Map Generalisation", working paper, *4th ICA Workshop on Progress in Automated Map Generalization*, Beijing, 2001, accessible at: http://www.geo.unizh.ch/ICA/docs/beijing2001/papers01.html