Generalization Operations and Supporting Structures

Kate Beard
Department of Surveying Engineering &
National Center for Geographic Information and Analysis
University of Maine
Orono, ME 04469
BITNET: Beard@Mecan1

William Mackaness
National Center for Geographic Information and Analysis
Department of Geography
State University of New York - Buffalo

ABSTRACT

Current GIS do not support wide flexibility for the performance of map generalization operations so users have limited opportunity for creating views of data at different levels of resolution. This paper describes a context for computer assisted generalization and reports on a set of generalization operators. The generalization operators are embedded within a larger scheme for a map design system which could be attached to a GIS. The selection and sequencing of operations is not fully automated but relies on user interaction. This approach is adopted to allow users maximum flexibility in tailoring maps to their individual needs. The system, however, is designed to provide substantial support for the user in negotiating this process. The final section of the paper describes data structures for supporting the operations within the context of this interactive environment.

INTRODUCTION

In many studies or projects, we wish to see some piece of geography represented or displayed in a simpler or more abstract form. We may also at any time wish to change the level of detail or level of abstraction of a representation. Although the ability to change the resolution of spatial or non-spatial information in a representation is highly desirable, this capability is not well supported by current GIS. Most commercial GIS software packages support generalization as one or two algorithms for line simplification (João 1990). These systems can be tricked into performing other generalization functions (Daly 1990), but the capabilities are not explicitly documented such that they are readily available to the casual user. The need for flexible and efficient changes in resolution warrants an expansion of generalization capabilities which are easy and intuitive for users to employ. Mackaness and Beard (1990) describe a user interface concept for a map design and generalization system. This paper expands on this earlier concept but focuses more specifically on generalization operations to be included in the system, the context in which they are applied, and proposed structures needed to support them. The paper

begins with an overview of the system to provide a context for the generalization operations.

CONTEXT FOR THE GENERALIZATION OPERATORS

McMaster and Shea (1988) and Shea and McMaster (1988) consider the important questions of why, when, and how to generalize. Much of the motivation and selection of type and degree of generalization is driven by user needs and purpose. The remainder is dictated by graphic media and format. This section develops a context for when and how to generalize within the proposed system based on two controlling factors: the user and graphic constraints.

The proposed map design system

The system as proposed by Mackaness and Beard (1990) assumes a vector GIS database exists. Characteristics of this database are described in greater detail in Section 4. It further assumes that users will interact with the database to select and extract information to composé views of the data at different levels of resolution or detail. Generalization operations in this case do not create new databases at coarser resolutions, but create materialized views of the original database. Views have been described in the database literature as an interface between a user (or application) and the database which provides the user with a specific way of looking at the data in the database (Langerak 1990).

In this system, we embed generalization operations within the basic functions of map composition and design. As itemized by Keates (1988) these include

- selection of geographic area,
- selection of information content,
- specification of format,
- specification of scale and
- specification of symbols.

These functions are intricately linked, but not necessarily in sequential order. Although at the outset one would most logically begin with selection of a geographic area, specification of the remaining functions could occur in any order including the ability to revise the size and configuration of the geographic area.

Full automation or system specification of these variables is probably not practical. Map design and generalization decisions depend largely on knowledge of map purpose so user interaction is highly desirable if not required. As Turk (1990) points out, improvements in human computer interaction will require shared cognitive responsibility between operator and computer. The proposed system therefore supports a high degree of user interaction, but is designed to assist the user in navigating through the process. The balance between user specification and system support is based on a consideration of which functions are best handled by the

system, which by the user, and which in some supportive arrangement between the two.
Figure 1 provides an overview of the system with an indication of which steps are user controlled and which are shared or managed by the system. Figures 2a-d illustrate user interface design for specification of the functions shown in Figure 1.

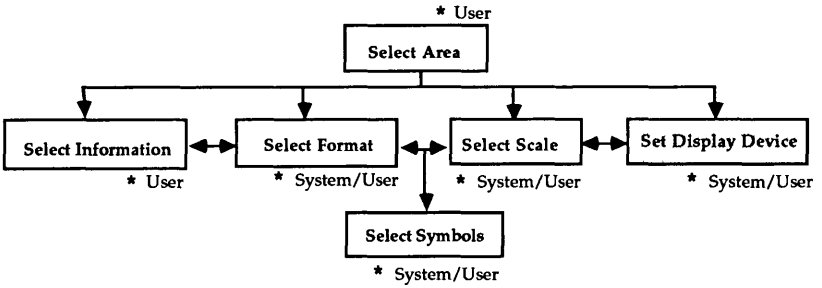


Figure 1. Overview of the system showing relationships between map design functions. Asterisks indicate functions which are controlled by the user and/or the system. There is an implied order to the functions given by the tree structure but the arrows indicate an ability to move freely between the various functions.

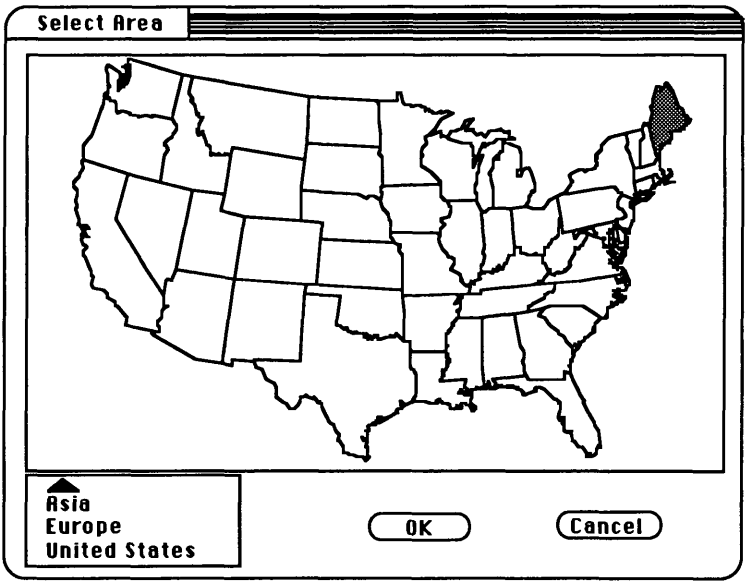


Figure 2 a. Illustration of user interface for selecting geographic area.

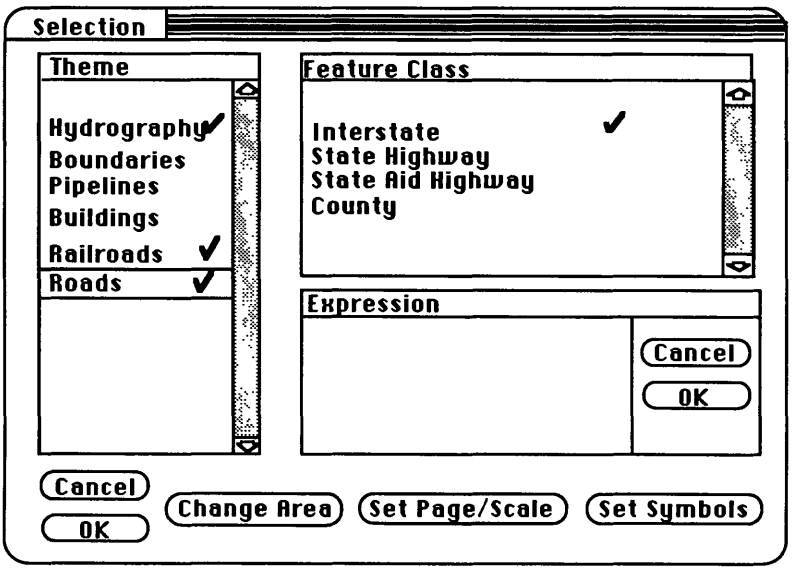


Figure 2b. Illustration of the user interface for selecting information content for inclusion on a map. Buttons on the bottom allow users to move to the other functions/menus.

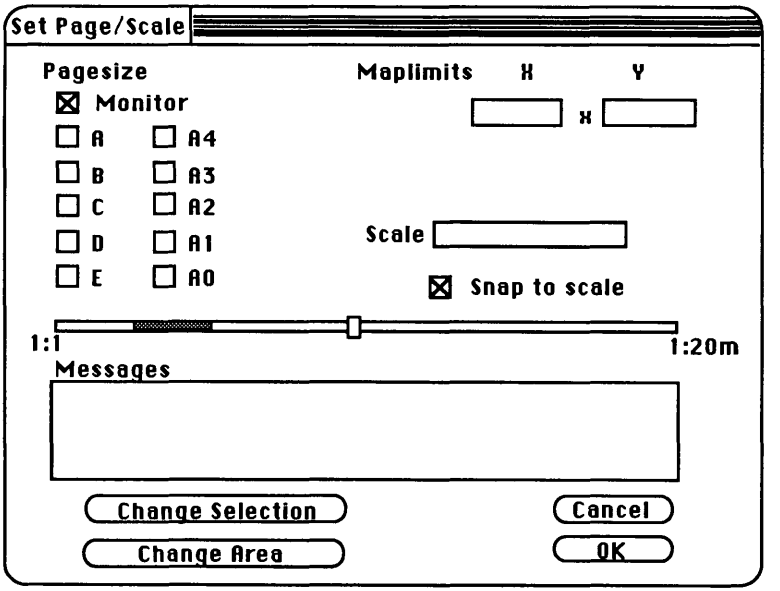


Figure 2c. Illustration of the user interface for selecting scale and/or page format. Users can move to the area selection menu or the information content selection menu from this screen.

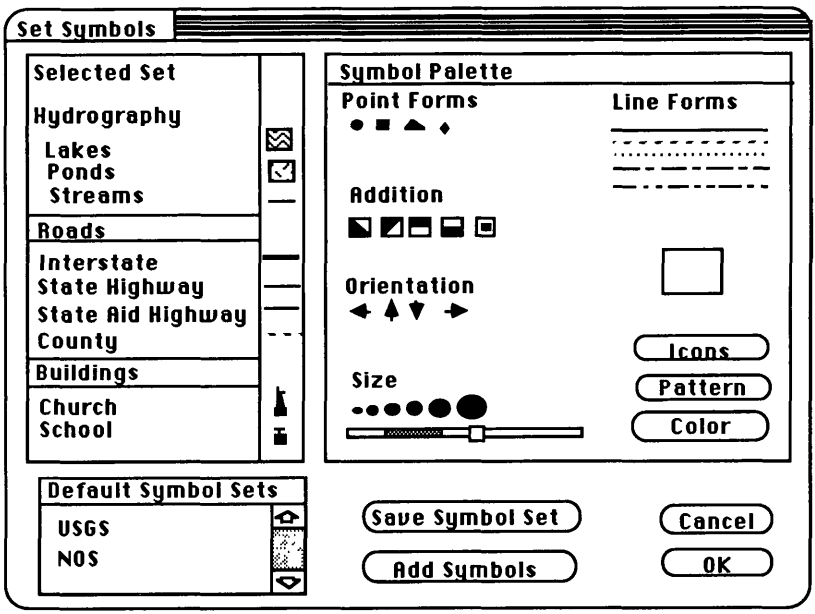


Figure 2d. Illustration of the user interface for specifying symbology. The selected information content is displayed on the left, and a symbol palette for making symbol choices appears on the left. The system indicates an appropriate range of dimensions for symbols given a scale. Selection of a default symbols set is also possible.

Once the user has made preliminary selections, the system can build on this information to provide clues and recommendations for subsequent steps. For example, if a user selects a geographic area which is 4 by 5 miles and selects, as a format, E size paper with a map area of 20 by 24 inches, the system computes a scale. As illustrated in Figure 2c, a computed scale would appear in the scale box and an appropriate scale range would be indicated by the shaded area on the slider bar for selecting scale. Alternatively, if the user specifies an area, information content, and scale, the system can recommend a range of appropriate formats. User specification and system feed back iterate toward an eventual result which meets users requirements and assures a legible display.

Specification of a geographic area, information content, format, scale, and symbology sets the scene for generalization. The combined specification of these five items can generate spatial conflicts or graphic interference, and to create useful and legible products these conflicts must be avoided or resolved. Conflicts can be avoided by re-specifying any one or more of the functions just described or resolved by generalization. In this paper we focus on resolution of conflicts by generalization operations.

Context for identifying and resolving conflicts

The types of conflicts which occur in map design are related to minimum requirements for maintaining graphic clarity and legibility. These

minimum requirements have been generally well documented in cartographic texts and cartographic production specifications. They are based on avoiding:

- areas which are too small
- line segments which are too short
- items which are too narrow
- items which are too close.

Items being too close results in congestion, coalescence, or conflict. The result of items being too small is imperceptibility and the same applies to segments which are too short and items which are too narrow. Congestion, coalescence, conflict, and imperceptibility are conditions described by Shea and McMaster (1989) that require some type of generalization for resolution.

These minima can be fixed as thresholds in any appropriate display units (eg. inches as shown in Table 1). Given a specified scale, format, and symbology, items selected from the database for display are screened against these thresholds to identify and locate conflicts. These conflicts are the minimum set of items or features which must be generalized. If any of the specifications are revised, the set of features which must be generalized will change

Conflict	Threshold
Too small	.01 sq. in.
Too short	.08 in.
Too narrow	.15 in.
Too close	.20 in.

Table 1. Illustrates fixed thresholds for legibility. In map construction these are transformed according to the selected scale and compared against dimensions of objects in the database.

Assume now the system has identified a list of all features and locations which are: too small, too short, too narrow, and/or too close (includes areas of overlap and coincidence). The specific function of the generalization operators is to resolve these identified problem areas. A set of rules could be formulated to direct the selection and application of generalization operators, but as generalization is intricately tied to map purpose, appropriate operations are difficult to anticipate for all cases. A simple rule would be to omit all areas which are too small. The user, however, may not wish to omit all small features, but exaggerate some or merge them with other nearby objects.

If the desired result is to be achieved from the users perspective, the user must have some involvement in orchestrating the operations. This prompts another balancing of tasks between operator and computer. In this case users are allowed to freely apply operators as they chose, but the system directs them to areas requiring generalization. An important function of the system is to clearly display all conflicts to the user and indicate when they have been resolved. This is handled by two methods. One is by listing objects which are in conflict with themselves or one or more other objects. The other is through graphic display of the conflicts. In the graphic display, all items in conflict (those falling below the thresholds) are displayed in red. All features which can be legibly displayed appear black. The items in red are the conflicts which must be resolved. As conflicts are resolved by generalization operations they are re-displayed in black and their resolution is also indicated on the corresponding tabular listing. Figure 3 provides an example of the interface for displaying conflicts to the user.

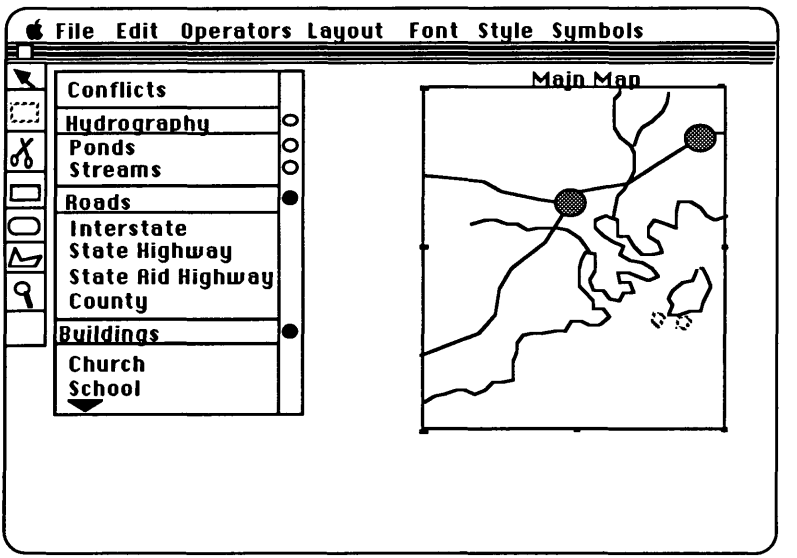


Figure 3. Example of the interface for displaying conflicts to the users. Conflicts are graphically (spatially) identified in red (dashed here) on the map as well as in the listing on the left.

GENERALIZATION OPERATORS

The function of the generalization operators in this context is to adjust a selected set of objects such that they can be legibily displayed at a specified scale, format and symbology. This section identifies a set of proposed generalization operators. Several cartographers have generated comprehensive lists of generalization processes (Steward 1974, Brassel 1985, McMaster and Monmonier 1989, McMaster 1990). Using these inventories, we can apply a structure to assist in identifying an appropriate set of operators. This structure distinguishes between operations on

graphic symbols and operations needed to simplify digital representations. These are referred to as structural operators: those that simplify or abstract the level of detail, and display operators: those that adjust the graphic display to ensure legibility. Structural operators can be seen to perform three basic operations: reduction in the number of objects, simplification of spatial detail, and simplification of attribute detail, with combinations of the three possible (Beard 1990). Display operators include operations such as displacement, masking, and symbol changes needed to resolve symbol collisions when a representation is displayed. This structure can be applied to McMaster's (1990) list of operations, for example, to assemble a toolbox of operators. For this system, operations from each category are selected to provide users a range of options and tailored for the purpose of resolving the conflicts identified above.

Proposed generalization operators
In this system we include the following operators:

Operations which reduce the number of objects
- select
- omit

Spatial operators
- coarsen
- collapse
- combine

Attribute operators
- classify

Display operators
- exaggerate
- displace

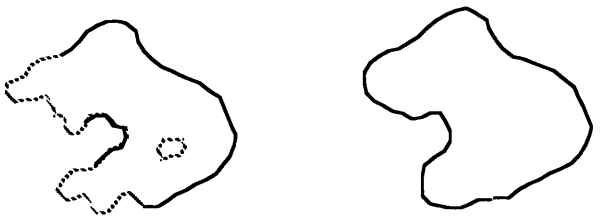
The names of many of these operators have appeared in the literature previously (Shea and McMaster 1989, Nickerson and Freeman 1986, Brassel 1985, Lichtner 1979), but their functions may differ here to specifically respond to conflict resolution. The functions of these operators as used in this system are described below.

SELECT: This is a special operator which must precede all others. It is required to initialize the composition of a graphic view of the database which can then be displayed on a monitor or as hardcopy output. The user is informed of information stored in the database and from this they may select items by theme, feature type, or instance (see Figure 2b). This operation allows the user to explicitly choose only desired items. For example, the user may select the theme roads, in which case all roads in the selected geographic area will be extracted for display. The user may also be more specific and select only Interstate Highways or to be most specific, select only Interstate 95 for example.

OMIT: Once items have been selected for display, the omit operator allows removal of objects. These objects are only removed from the display list and not from the database. As with the SELECT operator, individual objects may be removed or objects may be removed by theme, feature type,

or conflict type. For example the OMIT operator could be used to remove all objects which were too small.

COARSEN: This operator removes fine spatial detail (crenellations from a line). This operator could be applied to objects stored in the database with a high level of spatial detail, and which the user wishes to display in less detail. This operator works primarily on metric detail, but may change the topology of objects. Figure 4 illustrates an example of application of this operator to a lake with an island. In the resulting figure, the metric detail has been modified and the island has been removed, changing the topology.

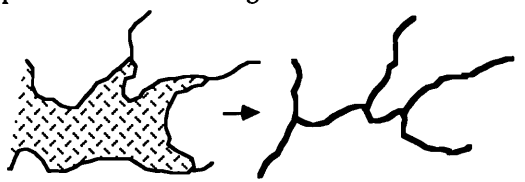


4a 4b

Figure 4a. shows a lake with an island at the level of detail it is stored in the database. Figure 4b show the same lake after application of COARSEN. The areas in conflict are show by dotted line (in color, these would be shown in red). In the resulting figure the conflicts have been resolved.

The user need not specify parameters for this operator. They only need select the object or objects to be coarsened and apply the operator. The operator uses the minimum thresholds which have been computed for the selected scale or format. The resulting representation is therefore appropriate to the selected scale. As shown in Figure 4, the small bays and island which fall below the threshold for areas too small, items too close, or items too narrow are removed by the coarsen operator. This operator can be applied to individual objects, themes or feature types.

COLLAPSE: The collapse operator substitutes a 1D or 0D representation for a 2D representation. This operator could be applied to objects stored in the database as areas, but which a user wishes to display as points or lines. Figure 5a and 5b show examples of COLLAPSE as applied to an estuary and a city. This operator must be preceded or succeeded by a symbol change. COLLAPSE resolves the legibility problem of items being too close, or COLLAPSE followed by a change in symbol width could resolve the problem of items being too small or too narrow.



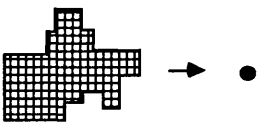


Figure 5a. COLLAPSE applied to an estuary

Figure 5b. COLLAPSE applied to a city.

COMBINE: The combine operator simplifies a spatial representation by merging objects which are nearby in space into a single new object. For example a cluster of small islands may be combined to form a larger island. The operator applies only to two or more selected objects and the result is always one new object. Thus COMBINE is strictly a localized operator. This operation must be preceded or succeeded by the CLASSIFY operator so that the resulting object is properly identified. Figure 6a and 6b illustrate an example of COMBINE. COMBINE resolves items being too small or too close.

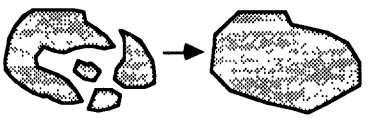


Figure 6a. COMBINE applied to islands.

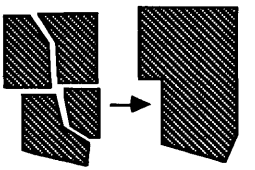


Figure 6b. COMBINE applied to fields.

AGGREGATE: This operator is similar to COMBINE but merges objects which are adjacent rather than those with intervening spaces. CLASSIFY must precede this operator as well. The aggregate operator can be applied globally by theme or by feature class.

CLASSIFY: This operator allows individual objects, feature types or themes to be assigned to a new class. The classification may be based on shared attribute characteristics of objects. The user or systems selects a set of objects and assigns a new class label (eg. For all objects with attribute D, Class = M). A symbol change must follow this operation, and when the new symbol is assigned, all objects assigned to the new class inherit the symbol. This operator does not directly resolve conflicts but is required as a supporting operation for operators which change the nature of an object (i.e. COMBINE and AGGREGATE).

EXAGGERATE: The exaggerate operator expands the size or width of objects. It can be applied by theme, feature type, instance or conflict type. The operator expands the object to meet the minimum threshold for legibility and therefore requires no parameter specification by the user. For a line or point representation, the width or radius is expanded. This can be accomplished by redimensioning a symbol. For an area, the operation performs a localized scale increase.

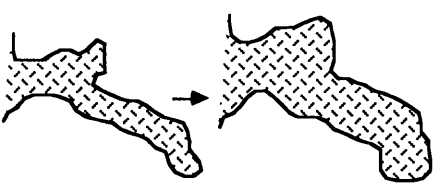


Figure 7a. EXAGGERATE applied to an inlet

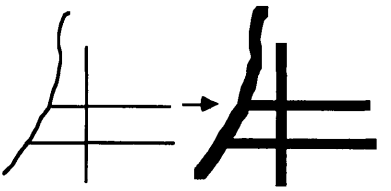


Figure 7b. EXAGGERATE applied to roads

DISPLACE: This operator is applied locally to two or more objects which are too close or overlapping.

Conflicts can be resolved by several different generalization operators with choice dependent on the desired outcome. Objects which are too small can be resolved by omitting them, exaggerating them, or combining them with other nearby objects. Objects which are too close can be resolved by omission, collapse, simplification, combination, or displacement. The selection and application of the operators is left to the user to allow them the most freedom in constructing a map to fit their needs. Some order is imposed in that some operators will not be accessible depending on the state. For example, SELECT is the only operator which can be accessed initially, and AGGREGATION may not be applied without first applying CLASSIFICATION.

Another key aspect in the design of operators is that they obey one overall rule. That is they are to resolve one or more conflicts when invoked and create no new conflicts. This rule is used to avoid convoluted iterations of operations in the resolution of conflicts. In particular this implies that all symbol specification occurs prior to generalization. For clearly, if symbol re-dimensioning occurs subsequent to generalization operations, new conflicts will arise and the generalization must be renegotiated.

SUPPORITNG STRUCTURES FOR GENERALIZATION OPERATIONS

For effective interactive use of the system, two tasks in particular must be performed efficiently. Conflict areas need to be identified rapidly so users can be quickly informed of the number and location of conflicts. Secondly the operators themselves must perform efficiently. In this section we consider supporting structures for facilitating each of these tasks.

Conflict Identification

Section 2 identified four types of conflicts. The first was areas too small to be legible. Identification of these conflicts is relatively straightforward. We first assume that areas are computed and stored as attributes of closed polygonal objects. Then, once a scale has been specified or computed, the minimum area threshold is derived, and conflicts are returned from the boolean function:

AREATOOSMALL = OBJECTAREA ≤ THRESHOLD

If AREATOOSMALL then DISPLAY (OBJECT, RED)

The number of comparisons required is of order N, the number of polygon objects selected for display.

Identification of the remaining conflicts depends on finding the euclidean distances within and between objects that are smaller than the minimum threshold computed from scale and symbol dimensions. This three additional boolean functions:

SEGTOO SHORT = SEGLENGTH ≤ SEGTHRESHOLD

WIDTHTOONARROW = OBJECTWIDTH ≤ WIDTH THRESHOLD

TOOCLOSE = POINTTOPOINT ≤ CLOSETHRESHOLD

To support these functions, we could conceivably pre-compute and store all distances between objects (objects in this case being points) as an ordered list. Discovery and retrieval of all violating objects could then follow by a search using distance as the key through the set of records ordered by distance between and within objects. This approach is sufficient to identify conflicts and provide the information to display conflict areas. The cost of computing and storing distances, however, is too high to justify simply the identification of conflicts. On the other hand if the cost can be spread over several other operations it becomes more justifiable. Our second criteria was to support efficient performance of generalization operations. In the next section we examine how pre-computed and stored distances figure into the resolution of conflicts and performance of the generalization operators.

Data Structures and Operator Performance

The number of operations dependent on knowledge of distance between objects implies the need for a database organized by spatial proximity. Such databases have been previously researched (Matsuyama 1984, Samet 1984) and arguments made for their use in the context of map design and generalization (Mackaness and Fisher 1987). Matsuyama's method, however, does not explicitly represent distance relationships among objects. Vornoi diagrams and the dual Delauney triangulation have also been proposed for representing spatial proximity relationships (Green and Sibson 1977, Brassel 1978, Gold 1987, 1989), but these also do not implicitly or explicitly store a full complement of distance relations. In Figure 8, form triangle edges we could derive distances from P6 to P4-P9 but not directly to P1 or P11.

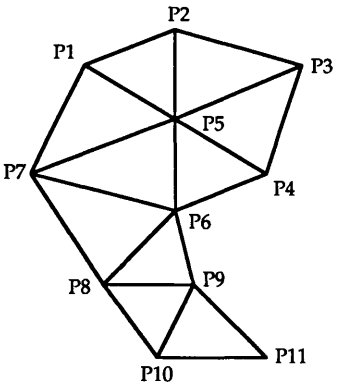


Figure 8. Distance relationships in Delauney Traingles.

In most cases, queries to these structures can return a spatial neighborhood or the set of objects within a neighborhood. Distances can then be

computed for these smaller sets. In this system, access to distance relationships is required frequently and uniformly over a geographic area. Given the level of interaction, the system also requires fast performance. The COARSEN, DISPLACE and EXAGGERATE operators in particular can benefit from immediate access to stored distance relationships. The next section describes a structure for storing and retrieving distance relationships. It assumes distances between points have been pre-computed.

A data structure for storing and retrieving distances.
Recall that the function of the generalization operators is to resolve identified conflicts and create no new conflicts. To assure that conflicts are resolved and no new ones created requires knowledge of distances between and within objects. Operators therefore need information beyond an ordered list of distances sufficient for identifying conflicts. In this case we need to know not just that a distance is sub-threshold but the locations where sub-threshold distances occur. The search condition thus involves the combination of three keys (Distance, X and Y), creating a multidimensional or range query problem. Assuming a threshold distance T, what we are after is a piece of the XY plane that yield clusters of points less that T distance apart.

The structure required is an indexed sequential data structure. Such a data structure accommodates both random and sequential access to records. In this case we adapt a method described by Orenstein and Merrett (1984). This involves interleaving bits of the tuple (DIST, X, Y) and storing the 'shuffled' tuples in the database. Interleaving the bits of a tuple maps a k-d space (3 in this case) to a 1-d space, creating a Z-ordering. The Z-ordering assures that points which are close in k-d space will be close in 1-d space. A similar ordering was first used by Morton (1966) for CGIS and has been replicated and expanded since by several others (Bentley 1975, Burkhardt 1983, Orenstein 1983, Ouksel and Scheuermann 1983, Tropf and Herzog 1981).

As Orenstein and Merrett (1984) note, the domains of the attributes in the tuple need not be the same size. An array [attr] can be used to indicate the attribute from which each bit was taken, yielding the shuffle function h(t) = [attr]{i} = i mod k where t is any tuple and k is the number of attributes per tuple.

Each bit in the 'shuffled' tuple corresponds to a split of a region of the three D space into two subregions of equal size. The bit equals 0 for one subregion and 1 for the other. Each additional bit splits the previous two subregions into two sub-sub-regions and so on. The direction of the split is given by the attribute [attr] from which the bit originated.

Sub-regions can be described by prefixes of the shuffled value. The addition of bits to the prefix refines subregions as described above. Smaller prefixes in other words correspond to larger pieces of the XY plane and larger distances.

Information retrieved from this structure can support identification of conflicts and provide direct input for the COARSEN, DISPLACE and EXAGGERATE operators. We discuss retrieval next in the context of the COARSEN operator.

Information is retrieved from the structure by a 3d search region SR. Initially SR is the entire space. A query region QR is posed given by the minimum bounding rectangle (MBR) of an object selected for COARSENing and by the threshold TOOCLOSE. If SR is outside QR, then SR contains no tuples satisfying the query and no action is required. If SR is inside QR, all points in SR satisfy the query and are unshuffled and returned. If SR overlaps QR but is not within it, SR is split into two new SRs. This step is applied recursively until SR is within QR. Several SRs may be required to cover a given QR , a weakness of this scheme which Orenstein and Merrett note. Once the final set of SRs is determined, tuples are actually retrieved by both random and sequential access. To use Orenstein and Merrett's notation $SR_{lo} : SR_{hi}$ denotes a range of shuffled values corresponding to a prefix. Retrieval of all points from an SR requires retrieving the tuples (t) such that $SR_{lo} \leq shuffle(t) \leq Sr_{hi}$. The data structure can be randomly accessed using SR_{lo} as the search argument. Then sequential accesses retrieve tuples until the shuffle value of a tuple exceeds SR_{hi}.

The set of tuples (DIST, X, Y) returned by this search procedure provide direct input for COARSEN. COARSEN performs a cluster analysis on the returned points and distances. The outcome of the cluster analysis is a reduction in the number of points such that no two are closer than threshold T. Simplified objects are then recomposed from the remaining points (see Figure 4).

A similar retrieval of records supports DISPLACE. DISPLACE is a localized operator applying to a small area. The area in which displacement will occur can be selected by clicking and dragging to define a rectangle. This rectangle and threshold TOOCLOSE define the query region QR. The search procedure returns the set of points within the rectangle and the distances between them.

SUMMARY

This paper discusses the context for a flexible and interactive approach to generalization. The design of the system seeks a balance between user responsibility and discretion and system intelligence to assist the user. The user makes initial selections for geographic area and information content. They may also specify scale, format and symbology or allow the system to compute or set defaults. Four types of graphic conflicts are identified as arising from these specifications. The selected objects can be too small, too short, too narrow or too close for the given scale and symbols dimensions. The purpose of generalization operators is to resolve such conflicts and assure a legible display. Identification and location of conflicts requires knowledge of distances between and within objects.

Distance computations are costly no matter how they are approached, but they are critical to operation of the system. The high degree of interaction demands high performance from the system. To support efficient interaction, we investigated methods for pre-computing and storing distances. An indexed sequential data structure is proposed to support efficient retrieval of information, but this must be subjected to testing to assure adequate performance.

ACKNOWLEDGEMENTS

This work has been supported in part by National Science Foundation grant SES-88-10917 and represents a contribution to Research Initiative 3, "Multiple Representations" of the National Center for Geographic Information and Analysis. The NSF support is gratefully acknowledged.

REFERENCES

Beard, M.K. 1990. 'Constraint Based Transformation for Map Generalization' NCGIA Symposium, "Towards a rule based symposium for map generalisation" Syracuse, NY, To appear in Map Generalisation: Making Decisions for Knowledge Representation London: Longmans (forthcoming).

Beard, M. K. 1988. Multiple representations from a detailed database: a scheme for automated generalization PhD thesis, University of Wisconsin, Madison.

Beard, M.K. 1987. How to Survive on a Single Detailed Database. Proceedings Auto Carto 8. pp. 211-220.

Bentley, J.L. 1975. 'Multidimensional binary search trees used for associative searching' Communications of ACM 18:9 pp. 9-517.

Brassel, K. and Weibel, R. 1988. 'A Review and Conceptual Framework of Automated Map Generalization,' International Journal of Geographical Information Systems. 2: pp. 229-244.

Brassel, K.E. 1985. 'Strategies and Data Models for Computer-Aided Generalization'. International Yearbook of Cartography. 25: pp. 11-30.

Brassel, K. 1978. 'A topological data structure for multi-element map processing', In Proceedings of the First International Advanced Symposium on Topological Data Structures for Geographic Information Systems (G. Dutton, Ed) VOL. 4. Addison-Wesley, Reading, MA.

Burkhardt, W.A. 1983. Interpolation Based Index Maintenance.' BIT 23:3 pp. 274-294.

Daly, R. 1990. 'Map Generalization using ARC/INFO' Research Report No. 10. Northwest Regional Research Laboratory, Lancaster University.

Gold. C. M. and Cormack, S. 1987. Spatially ordered Networks and Topographic reconstruction. International Journal of Geographic Information Systems. 1: pp. 137-148.

Gold, C.M. 1989. 'Spatial Adjacency: a General Approach. Auto Carto 9. pp. 298-312.

Green, P. J. and Sibson, R. 1977. 'Computing Dirichlet Tessellations in the Plane', Computer Journal. 21:2 pp. 168-173.

João, E. M. 1990. 'What experts systems don't know: the role of the user in GIS generalization'. Proceeding NATO ASI, on Cognitive and Linguistic Aspects of Geographic Space. Las Navas del Marques, Spain.

Keates, J. S. 1989. Cartographic Design and Production, New York, NY. Longman Scientific and Technical.

Keates, J. S. 1982. Understanding Maps London: Longman.

Mackaness, W. and Beard, K. 1990. Development of an Interface for user interaction in rule Base Map Generalization. Technical Papers GIS/LIS '90. Anaheim, CA. 1: pp. 107-116.

Mackaness, W. A. 1990. 'Application and evaluation of generalization techniques' NCGIA Symposium, "Towards a rule based symposium for map generalisation" Syracuse, NY, To appear in Map Generalisation: Making Decisions for Knowledge Representation London: Longmans (forthcoming).

Mackaness, W. A. 1988. Knowledge based resolution of spatial conflicts in digital map design Unpublished PhD Thesis, Kingston Polytechnic, UK. May 1988.

Mackaness, W. A. and Fisher, P. F. 1987 'Automatic recognition and resolution of spatial conflicts in cartographic symbolisation' Auto Carto 8 Baltimore, Maryland. pp. 709-718.

Matsuyama T, Hao, L.V. and Nagao, M., 1984. 'A file organisation for geographic information systems based on spatial proximity', Computer Vision, Graphic and Image Processing. 26:3. pp. 303-318.

McMaster, R. 1989. 'Introduction to Numerical Generalization in Cartography', In Numerical Generalization in Cartography. Monograph 40. Cartographica. 26: 1. pp. 1-6.

McMaster R. B. and Shea, K.S. 1988. 'Cartographic Generalization in a Digital Environment.: a Framework for Implementation in a Geographic Information System'. Proceedings GIS/LIS '88. San Antonio. 1: pp. 240-249.

Morton, G.M. 1966. 'A computer oriented geodetic database and a new technique in file sequencing.' Unpublished manuscript, IBM, Ltd. Ottawa Canada.

Muller J C 1989. 'Theoretical considerations for automated map generalisation', ITC Journal 3:4. pp. 200-204.

Nickerson, B. G. and Freeman, H. 1986. 'Development of a Rule-Based System for Automated Map Generalization,' Proceedings, 2nd International Symposium on Spatial Data Handling. pp. 537-556.

Orenstein, J.A. 1983. 'A Dynamic Hash File for Random and Sequential Accessing'. In Proceedings of the 6th International Conference on Very Large Databases. Florence, Italy. IEEE, New York, pp. 132-141.

Orenstein, J. A. and Merrett, T.H. 1984. 'A class of data structures for associative searching'. In Proceedings of the 3rd ACM SIGACT- SIGMOD Symposium on Principles of Database Systems. Waterloo, Ontario. ACM, New York, pp. 181-190.

Ouksel, M. and Scheuermann, P. 1983. 'Storage mappings for multidimensional linear hashing' In Proceedings of the 2nd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems. Atlanta, GA. ACM, New York, pp. 90-105.

Peckham, J. and Maryanski, F. 1988. 'Semantic Data Models,' ACM Computing Surveys. 20:3. pp. 153-189

Robinson, A. H. , Sale, R, Morrison, J. L. and Muercke, P. 1984. Elements of Cartography 5th Edition New York, NY. John Wiley and Sons.

Shea, K. S. and McMaster, R. 1989. 'Cartographic Generalization in a Digital Environment: When and How to Generalize', Auto Carto 9. pp. 56-65.

Steward H J 1974 'Cartographic generalisation: some concepts and explanations' Cartographic Monograph No 10 Toronto: University of Toronto Press.

Topfer , P. and W. Pillewizer . 1966, 'The Principles of Selection', The Cartographic Journal, 3:1. pp. 10-16.

Tropf, H. and Herzog, H. 1981. 'Multidimensional range search in dynamically balanced trees'. Angew Info. 2: pp. 71-77.

Turk, A. 1990. 'Towards an understanding of human computer interaction aspects of geographic information systems' Cartography (in press).