Algorithmica (2001) 30: 287–301
DOI: 10.1007/s00453-001-0011-0

Algorithmica

An Iterative Displacement Method for Conflict Resolution in Map Generalization[1]

M. Lonergan[2] and C. B. Jones[2]

Abstract. Cartographic generalization involves a trade-off between information content, accuracy and legibility. Conflict resolution, dealing with the problems of having too much information competing for too little space, is an important part of this process. For an iterative approach to conflict resolution two things are required: a measure of the acceptability of each intermediate map, and a strategy for finding a better one. Both the map quality measure and search strategy can have a large impact on the overall speed of the resulting process. This paper confines its map quality criterion to the minimum distance separating pairs of map features, an important component of legibility. This measure is combined with an iterative improvement technique, based on maximizing nearest neighbour distances, which attempts to find an acceptable solution where conflicts can be solved by displacement alone. The method also indicates those groups of features for which no such solution is possible. An experimental evaluation compares the method with one which uses simulated annealing and highlights its advantages with regard to generating many fewer candidate states and operating in a deterministic manner.

Key Words. Automated cartography, Generalization, Object displacement.

1. Introduction. Cartographic generalization is the process of selecting and representing information on a map in a way that adapts to the scale of the display medium and the purpose of the map. On all but the largest-scale maps, many of the symbols used to denote the mapped features will occupy a larger area on the map than a correct scale representation. The result is that there is a competition for map space requiring selectivity in the choice of which features to represent as well as symbolic abstraction of form. This can involve simplifying and caricaturing individual features as well as representing groups of related features with single symbols. It may also be necessary to displace nearby objects from their initial locations to ensure their legibility [1], [2].

Given the numerous ways in which information may be represented, some criteria are needed for determining whether a map can be regarded as acceptable for its intended use [3]. For the purposes of automation, map generalization may be regarded as an optimization problem. The criteria determining map quality can then be formulated as an evaluation function and used to compare different possible maps of a given situation. How such a measure of map quality is best defined will depend on the particular circumstances. The simplest definitions will produce a Boolean result; either the map is good enough or it is not. While this is sufficient for deciding whether a single map is acceptable, it gives

[1] This work was funded by EPSRC Grant GR/L49314 in collaboration with ESRI(UK).

[2] Department of Computer Science, Cardiff University, Cardiff CF24 3XF, United Kingdom.

Received December 21, 1998; revised November 26, 1999. Communicated by M. van Kreveld.
Online publication March 21, 2001.

no assistance in finding such a map. For this a measure is required that indicates how far from acceptability the map is. Information on how and where the map is inadequate is also valuable. A useful measure also needs to be unambiguous, objective and repeatable. Ideally it should be both good at discriminating between potential solutions and quick to calculate.

Once a quality measure has been chosen, it can be used to evaluate potential maps. However, there is an infinite number of possible ways of representing even a small data-set. In general, most real problems involve producing maps of relatively complicated situations from large databases. Therefore only a very small subset of the potential solutions can be examined individually. A strategy is needed to search the space of candidate maps. Without detailed knowledge of the topography of the search space, non-exhaustive strategies cannot be guaranteed to find an optimal solution. This means that realistically the search is for an acceptable, rather than the "best", map for a particular purpose.

The speed of a search depends on the strategy and quality measure chosen and how well they match the search space. For each individual search, the interaction of strategy and measure can be seen as producing an ordering on the search space. Each numeric quality measure can be considered to define a surface over the ordered space. The complexity of this surface's topography determines the overall speed of the search. The choice of an appropriate quality measure and search strategy can greatly increase the speed of the search. In the case of map generalization, good generalization methodologies begin by making those changes that are most likely to reduce the number and complexity of subsequent modifications required.

Two features on a map can be said to be in conflict if they are too close together to be distinguished. This is usually represented by a fixed distance corresponding to average visual acuity. A method for resolving conflicts by object displacement is recognized as an important part of map generalization. The problem of how to relocate single objects has been investigated previously, for example in [4] and [5]. Previous work on the iterative displacement of multiple objects has attempted to minimize the number of conflicts observed [6] or some function of the total amount of conflict observed [7]–[9]. Two basic strategies have been applied to find the new positions for objects. In [6] and [10] fixed sets of trial positions were investigated for each object. This project follows the more common approach of directly calculating displacements. The method used focuses on reducing the severity of the worst conflict and allows the creation of new conflicts, enabling the solution of conflict by the displacement of other objects not initially in conflict. The result is a continuous and deterministic gradient descent method applicable to a wide range of situations. It identifies the set of features whose modification or elimination could help solve each individual remaining problem on the map. The method used to calculate displacement vectors also has the advantage of reducing the number of object displacements to be investigated when compared with the simulated annealing method of [6].

The paper describes a simple measure of map quality and an iterative strategy for object displacement that can be used in the search for an acceptable map of a given situation. The computational complexity of the new method is evaluated. An experimental implementation is then compared with one using simulated annealing. Finally, the potential of this approach for practical map generalization is discussed.

2. Measuring Map Quality. A very simple measure of overall map quality is used. We consider a map to be made up of a collection of rigid polygonal objects representing individual features. In principle each one could either be included in the final map or excluded from consideration, though this paper only considers improving the map by object displacement without deletion. Features must be separated from each other by at least a given minimum distance in the final map and each one can be moved by up to a given maximum displacement from its original, reference, position to achieve this. A map that satisfies these criteria is considered acceptable. For intermediate maps all the features must be within the given distance of their reference positions. The better of two intermediate maps is then the one with the larger minimum nearest neighbour distance. (In the case of a tie the second smallest nearest neighbour distances are compared, and this process repeated until a difference is detected.) This quality measure is incomplete in that it considers only the issues of legibility, as affected by separation distances, and locational accuracy. A complete measure of map quality would also have to consider many other factors. Investigating the limitations of the results produced by using this measure may help identify necessary modifications. For now we are attempting to find a map that is acceptable under these criteria and contains as much information as possible.

3. Search Strategy. Starting from an initial map with each object located at its reference position, each object in turn is considered. Any object that is in conflict with any other is moved in an attempt to maximize the distance from it to all its neighbours, subject to obeying the maximum displacement constraint. This process is repeated until the displacements made become small. During the first pass through the process all objects have to be considered. After this it is only necessary to look at those whose surroundings have altered since they themselves last moved. If the final state does not satisfy the minimum separation criterion, it will result in groups of objects that violate the criterion. The term cluster is used for these groups. The nearest neighbour of each object within a cluster is also a member of that cluster. A path can be traced between any two members of a single cluster consisting of a series of steps between cluster members such that none of the steps is greater than the minimum separation distance. Only pairs of objects that both belong to the same cluster are less than the minimal distance apart.

If any clusters of objects in conflict exist, it will be necessary to modify at least one member of each cluster in order to produce an acceptable map. This could involve the deletion or merging of objects or changes in their representation. By repeating such a cycle of cluster identification and object modification, all inter-object distances could be made acceptable, though this paper does not address the implementation of such a procedure.

An exact solution of the problem of maximizing the distance between irregular polygons is computationally expensive. In cartographic generalization the situation is further complicated by the requirement that no polygon moves further than the predetermined maximum displacement distance. This is a more complicated form of the problem of placing rectangular text labels to identify point features, which has been shown to be NP-hard [11]. A simple approximation to the problem was therefore used. The shortest vectors connecting the individual edges of an object to each of the edges of all nearby objects are identified, and a selected subset of these used to calculate a new position for

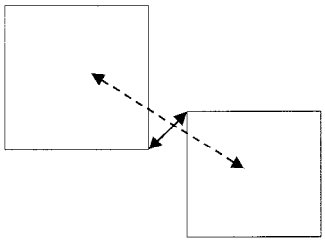

Fig. 1. Two objects in conflict. The shortest displacement that will solve this is horizontal. Motion along the line connecting their centres (broken arrow) will preserve the apparent relationship between their positions. The solid arrow shows a possible interaction between edges that is almost perpendicular to these preferred directions.

the object under consideration. The displacement process is repeated for each object in turn.

The calculation of conflicts between individual edges, rather than just between whole objects, is important in certain situations. Without this, the displacement of an object away from one part of a large and irregularly shaped neighbour may bring it into conflict with another part. However, if all edges are allowed to interact other problems can result. Figure 1 shows a simple case where this happens. The solid arrow shows the possible interaction between the lower edge of the upper object and the upper edge of the lower object. A large displacement will be required to separate the objects in this direction and, could even, in some circumstances, result in the right hand, originally lower, object rising up to be level with or even above the left hand one. A smaller horizontal displacement will achieve the same overall separation. Displacement along the direction indicated by the dashed arrow, while needing to be slightly greater than the horizontal amount will best preserve the relative positions of the two objects. Such displacements can be found simply by defining a "field of influence" for each edge.

4. Fields of Influence. Point features and those made up of a single line segment can each be treated as a unit for the calculation of their contribution to displacement. For areal features only edges that face each other are considered to interact. Each edge is then considered to have a "field of influence", within which it can interact with other objects. The field of influence extends to a distance of twice the minimum object separation r, from both sides of the edge, to allow for maps containing nested objects. For an isolated edge the field of influence includes all space within distance r of the edge. This region is referred to as B. When an edge is connected at each end to a neighbouring edge, the field of influence is in two parts, one for each side. The field of influence on each side of the edge is found from the intersection of B with a semi-infinite wedge, bounded on three sides, by the edge and by two rays emanating from the vertices of the edge. If the angle between the edge and its neighbouring edge is less than 180°, the ray for the respective vertex is coincident with the neighbouring edge. If the angle is greater than 180° the ray bisects the angle between them. If there is no neighbouring edge the field of influence wraps round the vertex. In Figure 2 the edges of an object are shown as solid lines, and the boundaries of the field of influence of one of the edges (marked e in the diagram) are

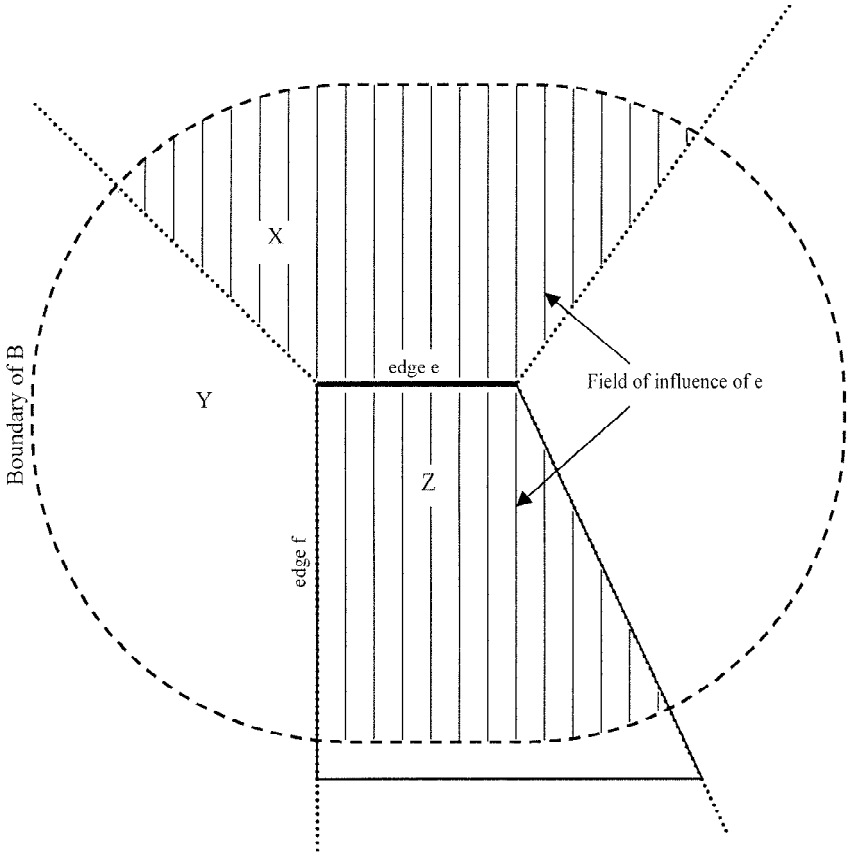

Fig. 2. The "field of influence" of an object edge. Only features located within the shaded area (for example at positions X and Z, but not Y) can be influenced by edge e of this object.

indicated by broken lines. An object at position X can only be affected by edge e; at Y only by edge f; and at Z by all four edges of the object shown.

For each edge on the original object the shortest vector between it and each edge of each neighbouring object is found. If this lies entirely within both edge's fields of influence they are considered to be able to affect each other. The force, $\mathbf{p}_{j,k}$, exerted by edge e_j on edge e_k, the shortest vector between which is $\mathbf{v}_{j,k}$, is then given by

$$\mathbf{p}_{j,k} = -(2s - |\mathbf{v}_{j,k}|)\mathbf{v}'_{j,k},$$

where s is the minimum acceptable separation distance, $|\mathbf{v}|$ is the magnitude of vector $\mathbf{v}$ and $\mathbf{v}'$ is the unit vector parallel to it. The factor 2 in the equation speeds up the handling of clusters, and is discussed later in the paper.

5. Displacement Calculation. For each object that is in conflict with some others, all the forces on it need to be combined to give a single displacement. Hirsch [12]

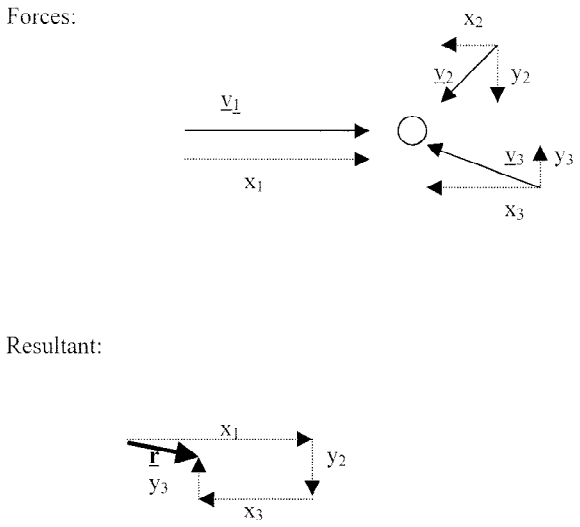

Fig. 3. Method used for combining forces on an object into a resultant. The forces are resolved along two standard directions and the largest components in each of the four cardinal directions are summed to give the resultant.

applies a similar, vector-based, approach to the name-placement problem, though they sum the repulsive vectors. Summing or averaging can result in several small effects combining to outweigh a single larger one. This is avoided here by resolving each individual force along the x and y (or east–west and north–south) axes and summing the single largest component in each of the four directions to produce a resultant force. Figure 3 demonstrates this for the three vectors $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ (shown as solid lines). Each is resolved into components in the x and y directions (the dotted lines). The largest positive and negative components for each axis (in this case x_1, x_3, y_2, y_3) are then summed to give the resultant, $\mathbf{r}$, which equals $(x_1 - x_3, y_2 - y_3)$.

Displacing the object by a vector parallel to this resultant, and with half its magnitude, should place the object at a point where the new forces it experiences are in balance. (The use of fields of influence that reach out to twice the minimum acceptable separation distance, as described above, ensures the satisfactory displacement of objects whose motion would otherwise be unopposed in one direction.) The object is then moved by this amount, assuming that this does not take it too far from its reference position. If this is not possible, it needs to be placed at the best of the positions that meet the object displacement constraint. Since the purpose is to maximize nearest neighbour distance this is the location within the permissible area that minimizes the largest remaining force on the object. Figure 4 shows how this is calculated. While the description of the process is long winded, the calculations it requires are simple and straightforward.

In Figure 4 the reference position of the object is at O and the calculated displacement would move the object from some point P to place it at T. Depending on the relative sizes of the forces involved and the maximum allowable displacement, there are four possible cases. Each of the four circles shows the area within which the object may be placed for

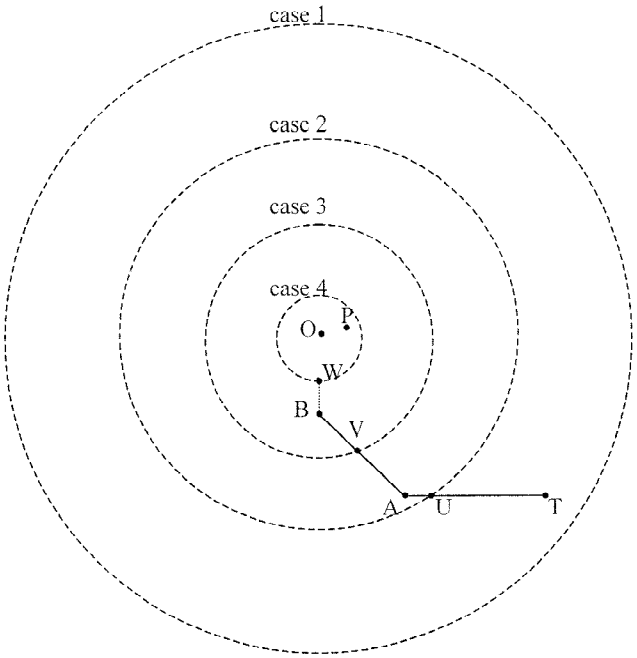

Fig. 4. Movement of an object. Where the preferred displacement to T would take the object too far from its original position a smaller displacement is made. As described in the text, each circle corresponds to a different maximum allowable displacement of an object originally located at O. T, U, V and W are the final positions of the object under the four cases.

one of the cases. The radius of each circle corresponds to the maximum displacement, m, for the object on a particular map at a particular scale and contains the set of acceptable positions for it. Here acceptable means only that the object is sufficiently close to its original position and does not consider any potential conflicts with other, neighbouring, objects. At the start of the process the object is at some point P. If the object has not been displaced previously this will be coincident with O, otherwise it may not be. Conflicts with neighbouring objects can be resolved to give forces on the object of $+a$, $-b$, $+c$, $-d$ along the positive and negative x and y axes, respectively.

Given the forces at one point, those at other, nearby, locations can be approximated by taking each resolved force to reduce by one in magnitude when the object moves a unit distance in the direction of the force. So if the object is placed at the point one unit below and two to the right of P the four forces become $(+a-2, -b-2, +c+1, -d+1)$, using the same ordering as above. These changes are not permitted to alter the directions of the four forces. Where this would occur the relevant forces are instead set to zero. The result is approximate in that it treats the surrounding objects as if they formed a box aligned with the axes and containing the object being displaced, and assumes no other objects can become involved in the process during the displacement.

T is the point at which the above approximation produces a zero overall resultant force on the object. It is displaced from P by the vector $\frac{1}{2}(a-b, c-d)$ since this produces a

balance in both the horizontal and vertical forces. This is therefore the preferred position for the object to move to. However, this may not be an acceptable position for the object, since it may require moving it further from its initial position than is permitted.

Case 1 is the simplest. Here T is within the set of acceptable positions for the object, so it moves to this point. The outer circle in Figure 4 corresponds to this situation.

If T is an unacceptable position for the object with regard to its displacement, the object must instead be placed in the acceptable position that maximizes its nearest neighbour distance. Though there is no net force on the object at position T, the magnitudes of the component forces at this position may not be zero. A is the point at which the three largest forces on the object are equal to the largest forces at T. It is found by considering the magnitudes of the component forces at the position T. Depending on the relative sizes of the horizontal and vertical forces, A will be aligned either vertically or horizontally with T. Assuming that the vertical forces are greater than the horizontal ones at T, and T lies to the right of O, as shown in Figure 4, A is displaced from T by $((a+b-c-d)/2, 0)$. At this location the object would experience three forces of magnitude $(c+d)/2$. The fourth would be the greater of $(a+b-(c+d)/2)$ and zero.

Case 2 occurs when part of line TA is within the area of acceptable positions for the object, but T is not. The object is then placed at the acceptable position on line TA that is closest to T. This is marked as position U for the second labelled circle in Figure 4.

If no part of TA is an acceptable position for the object, the object must experience a larger maximum force at the best of the acceptable positions than it would at T. This is found by projecting a line angled at 45° to the horizontal and towards O. B is the first point where one of the line's coordinates matches that of O. For clarity this is again only shown for one possibility.

Case 3 occurs when B is an acceptable position for the object but A is not. The object is then placed at point V, the point on AB that is closest to A and acceptable.

Case 4 is where point B is unacceptable. The object is put at position W, the acceptable point on OB that is closest to B.

This process allows a reasonable approximation to the best location to be found. It reduces the number of object movements that are required while avoiding having to recalculate all the forces for each intermediate step.

6. Implementation. The system was implemented in C++. A simple data structure with individual objects stored as polygons containing pointers to their edges and vertices was used. Each edge points to its endpoints and each point stores its own location. To speed up the system's operation an initial preprocessing step identifies objects that have acceptable positions with regard to their own displacement but which might be too close to some other objects. During the operation of the main program, another check is made to find whether the objects' centres are close enough together for conflict between them to be possible in their present positions. The overall structure of the system, as presently implemented, is shown below:

1. Preprocessing:
   (a) Load original objects, O_i. These remain unchanged through the process and act as a reference for calculating object displacements.

(b) Make a copy of the objects, M_i. These "map objects" can be moved and may appear on the final map.
(c) Find centroid (average of vertex positions) $\mathbf{c}_i$ and spread (distance of furthest vertex from centroid) r_i of each map object M_i.
(d) Find all pairs of objects M_i, M_j that satisfy

$$|\mathbf{c}_i - \mathbf{c}_j| < (r_i + r_j + 2m + 2s),$$

where m is the maximum displacement allowable and s is the minimum separation distance required. (No other objects can conflict for any allowable positions.)
(e) Make all map objects modifiable.

2. Find all those of the pairs that satisfy step 1(d) that also satisfy

$$|\mathbf{c}_i - \mathbf{c}_j| < (r_i + r_j + 2s).$$

These might conflict in their present positions.

3. For each map_object M_i that is modifiable:
(a) set max_force $= 0$;
(b) for each edge e of M_i:
for each edge e' of each object M_j that might conflict with M_i in its present position:
if e' is in the field of influence of e, and e is in the field of influence of e' (see Section 3 for description of these):
(i) Find shortest vector $\mathbf{v}$ between e and e'.
(ii) Calculate

$$\begin{aligned} \text{force}(e, e') &= (p_x, p_y), \\ \text{force}(e, e') &= (2s - |\mathbf{v}|)\mathbf{v}' \end{aligned}$$

($\mathbf{v}'$ is the unit vector parallel with $\mathbf{v}$).
(iii) If $|\text{force}(e, e')| >$ max_force;
set max_force $=$ force(e, e');
(c) if |max_force| $> s$ (M_i is in conflict with at least one other object)
(i) Find maximum and minimum of each of p_x ($p_{x\max}$, $p_{x\min}$) and p_y ($p_{y\max}$, $p_{y\min}$).

$$\begin{aligned} a &= \text{Max}\{p_{x\max}, 0\} \\ -b &= \text{Min}\{p_{x\min}, 0\} \\ c &= \text{Max}\{p_{y\max}, 0\} \\ -d &= \text{Min}\{p_{y\min}, 0\} \end{aligned}$$

(ii) Calculate total_force$(i) = (a - b, c - d)$.
(iii) If moving M_i by a distance equal to total_force$(i)/2$ is permissable, do this; else move M_i to best acceptable position (as described in Section 4).
(iv) Make M_i unmodifiable. If M_i has moved make all its neighbours modifiable.

4. Repeat from step 2 until maximum displacement is less than some preset value.

7. Time Complexity. The system falls naturally into two parts for analysis: preprocessing and displacement. Four attributes of the data determine the time the system takes

to calculate results: the total number of objects considered, n; the maximum number of edges any one object has, e; the maximum number of other objects that can directly influence each object, d; and the maximum number of objects in a single cluster, c.

The preprocessing stage described (step 1 above) has a worst-case time complexity of $O(n^2)$, though this could be improved by the use of an appropriate spatial data structure. Since the neighbourhood search procedure is based on a fixed maximum distance relative to object edges, following [13] in the use of a triangulation of the object vertices and edge subdivision would provide an $O(n \log(n))$ method.

The maximum time required to carry out one displacement of each object (steps 2 and 3 above) is $O(nde^2)$, since each edge of each object has to be checked against all the edges of all the objects that can directly interact with it.

The number of iterations through the process (step 4) depends on the cluster size. The factor 2 applied in calculating the individual forces on an object increases the displacement of objects under forces from only one direction beyond that necessary to resolve the immediate conflict they are experiencing. The overall effect of this is to tend to displace the outermost objects in a potential cluster to their extreme positions in the first iteration of the process. These movements produce enough space for further objects to move. Each subsequent iteration will then push another "layer" of objects out. Without this factor each iteration can do no more than halve the remaining conflicts.

Assuming an even distribution of objects, the total number of iterations required by the system will therefore be $O(c^{1/2})$, making the total calculation time for the whole system $O(nde^2c^{1/2})$. Ignoring map edge effects, it would seem reasonable that the cluster size should depend on d, but for an individual situation the actual relationship may depend on the objects' arrangement and the particular scale change considered.

8. Results and Comparison with Simulated Annealing. Figure 5 demonstrates the methodology for a simple test case. Here three moveable objects are partially constrained

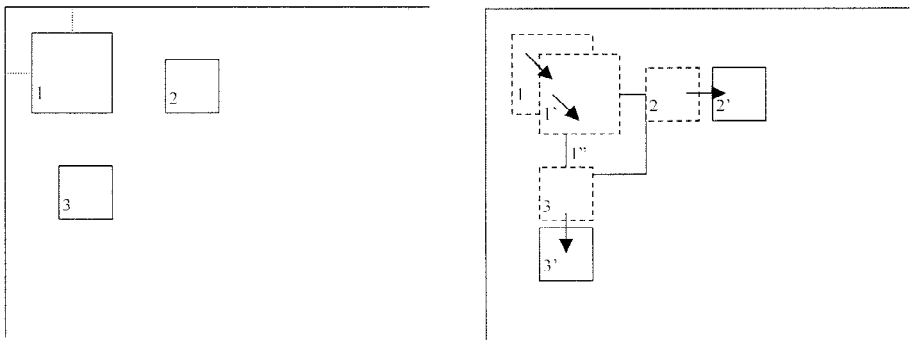

Fig. 5. Displacement of three objects near a fixed boundary. (a) Initial state. The broken lines indicate conflicts. (b) After displacement. $1''$, $2'$, and $3'$ are the final positions of the three objects.

by a fixed boundary. Initial conflicts are indicated by dotted lines in Figure 5(a). In this situation, objects that are not directly involved in any conflicts need to be displaced to make space for others to separate. The initial movement of the largest object (from 1 to $1'$ in Figure 5(b)), causes two additional conflicts with the other objects (2 and 3), leading on to their displacement (to positions $2'$ and $3'$, respectively). A further displacement of the large object (from $1'$ to $1''$) utilizes the extra space for the solution of the initial conflicts. This strategy is less easily applied to methods based on either the total amount of conflict (which is unchanged by the initial object displacement) or those that count conflicts. Simulated annealing can be used to deal with such situations [6], but this can require a large number of trial moves and produces a randomly chosen member of the set of possible solutions.

If the displacement of the smaller objects (2, 3) in Figure 5(b) were only enough to solve the immediate problem, the cycle of displacement would repeat indefinitely, halving the conflict each time round. Instead, as the objects are being pushed from only one side, they move as far as possible away from this force. The subsequent displacement of the larger object (to $1''$) will then produce a solution, assuming one exists. In this example the whole process makes four moves to produce an acceptable solution to the problem, one more than would be required if each object were placed directly into its final position, but without the necessity to consider multiple object interactions directly.

While the solution found is acceptable under our criteria, it is not ideal. A further post-processing step to draw each object back towards its reference position, where this is permissible, would improve the final position further.

A more complicated situation is shown in Figure 6. This involves many more ($\sim$ 320) moveable objects and more complex fixed linear boundaries. This is the same example as used by Ware and Jones [6]. Their approach considered a total of 350,000 moves to produce a solution. The new method makes around 400 moves, depending on the stopping conditions used. A total of 38 conflicts remain after this (out of an initial 153), similar to the number Ware and Jones [6] were left with. Measuring the total amount of conflict remaining in the map, by subtracting the lengths of the shortest inter-object vectors from the minimum acceptable separation and summing the results, gives an alternative measure of success. Our method reduces this from 578 to 66. As the technique operates by attempting to reduce the severity, rather than number, of conflicts this may be a better assessment of its achievements.

Figure 7 shows a portion of the area covered by Figure 6. This is the "Region 1" discussed in [6] where gradient descent appeared unable to solve all the conflicts observed. The broken lines indicate the original locations of the objects and the solid lines their final positions. The new quality measure has enabled our gradient descent method to resolve all the conflicts in this area, where [6] required simulated annealing.

While the number of moves made under our method is much lower than that for the simulated annealing approach, each move requires more complicated calculations. Direct comparisons of the speeds of the two methods are complicated by the different program structures and machines used. Ware and Jones [6] report a time of 40 seconds on a "Sun Enterprise 2 model 220 (2×200 MHz Ultrasparc processors)" following preprocessing to generate a triangulated data structure and store the initial values of all the object interactions. These then merely required updating as objects moved. The method described here required 32 seconds on a 233 MHz PC to carry out one pass

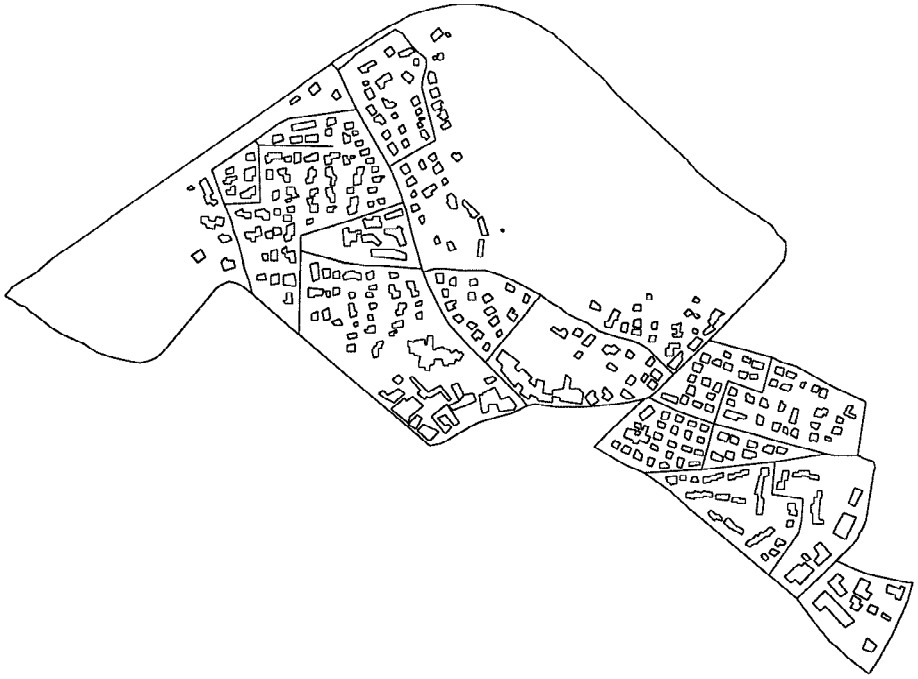

Fig. 6. Dataset used in the experimental evaluation of the method.

through the process and 78 seconds to carry out four passes and produce the results shown, again following preprocessing to identify each object's potential neighbours. Our program recalculated all the interactions at each step. The simulated annealing experiment considered only a limited set of potential positions for each object (though that is not a requirement of the method), rather than the continuous distribution in this paper. Neither system was optimized for speed of calculation.

9. Discussion. The method presented here provides a reasonably quick deterministic solution to the problem of finding a set of displacements that solve spatial conflicts between a group of objects. The clusters it produces also indicate which objects' modification or elimination could contribute to the solution of each part of the problem that is insoluble to displacement alone. It does this purely by looking at interactions between pairs of objects. Any iterative generalization technique has to balance the number of operations it carries out against their individual complexity. By avoiding the direct calculation of multiple object interactions this approach simplifies each step, while its combination of the effects of all the interactions of pairs of objects reduces to a manageable level the number of times each object is moved.

The method does not attempt to produce the best possible map of a situation, merely an acceptable one. The results could be further improved by post-processing to reduce the amount of unnecessary displacement of objects. Individual objects are deliberately

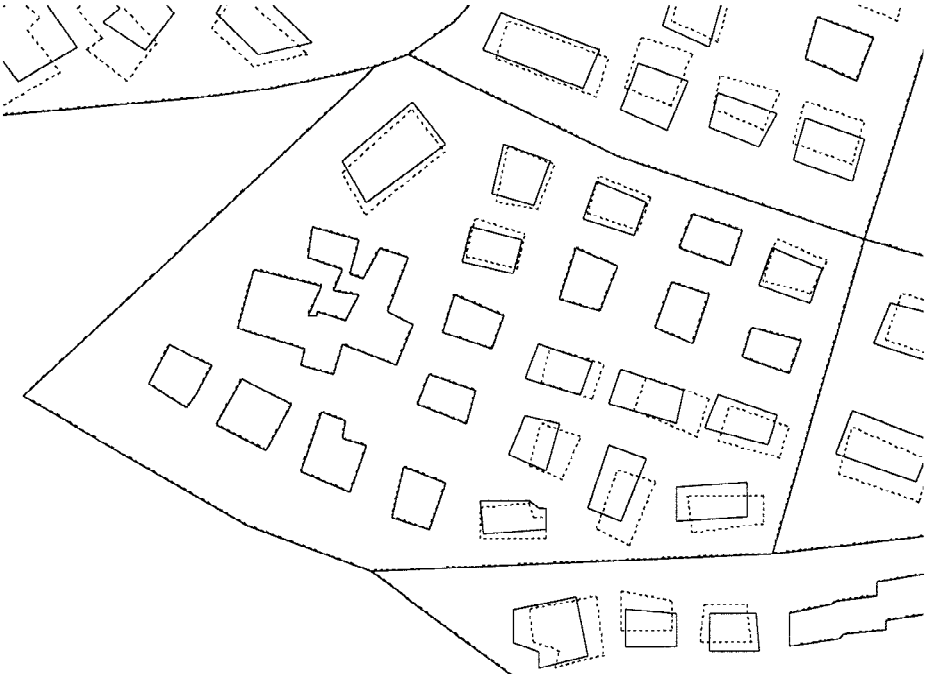

Fig. 7. Magnified view of part of dataset in Figure 6 showing results of displacement. Broken outlines indicate original positions, solid lines are final positions.

displaced further than is immediately necessary during the process to reduce the number of moves required to handle large groups in conflict. It would be possible to draw each object back as near to its reference location as is possible without causing conflicts with others, though whether the improvement in the results would justify the extra computation involved is unclear. Future work will examine this issue.

For simplicity, our implementation calculated the "forces" on, and displacement of, each object in turn. An alternative strategy, as used in [6], would be to store all the interactions between the objects, move the object under the greatest resultant force and then update everything this affects. This may well reduce the number of moves required to find a solution, but requires the storage and manipulation of much more information. For real situations, involving large numbers of objects and interactions, it is not clear whether this, along with the need to identify the next object to move, would outweigh any advantages of a more sophisticated strategy. This will also be further investigated.

Another possible way to accelerate the conflict resolution process is to carry out a simple preprocessing step to deal with some of the most obviously insoluble situations. The method we describe here involves detailed calculations, comparing individual edges of the objects. Less complex techniques, perhaps based on object areas and approximate arrangements, may suffice to identify a significant proportion of problems. The present implementation uses simple circle-based methods (steps 1(d) and 2 in the algorithm described) to restrict the numbers of potential interactions to be examined. These are

similar to the methodology in [10], though that paper applies them directly to the selection of new positions, considering the unnecessary displacements produced to be outweighed by the overall simplicity. What proportion of situations require which techniques is likely to vary between scales, data sources and map types, and only become apparent through experimentation.

A major advantage of this approach over simulated annealing is its determinism. A given initial situation will always produce the same result, while a system based on simulated annealing can produce very different results each time it is run. It is not always essential that two maps of the same situation and scale should be identical, but there are cases where it may be important. Another benefit of the new method is its identification of all possible objects that could be merged or eliminated to assist in solving a problem.

The methodology presented here is anisotropic. As the "forces" are resolved along the map's north–south and east–west axes, rotating the axes could produce slightly different results. Resolving along and perpendicular to the direction of the largest "force" experienced by an individual object could solve this, but makes the results very sensitive to initial conditions in cases where non-orthogonal forces are similarly sized. Whether the small differences this makes in the final map are important will, again, depend upon its purpose.

Further work is under way to extend the method to include generalizing lines and the displacement of deformable objects. Once these processes have been added to the system it will be possible to start examining the results for simple maps of real situations and assessing their utility.

Overall the relative crudity of the measure of map quality may limit the use of the method's results to certain purposes, but it may be that relatively simple modifications to the quality measure could greatly increase its applicability. Many maps allow at least some type of objects to be in direct contact and this may need to be incorporated into the technique. Representing large-scale structure and retaining patterns within groups of objects while removing some members is likely to be more difficult. Ruas [8] suggests an approach to the problem of local distortion, based on each object's motion drawing its neighbours along with it. Incorporating this into the method might provide a partial answer to the problems of large-scale structure. However, an entirely different approach may be required to identify and simplify regular patterns of objects and networks of linear features (such as those suggested in [14] and [15]). The approach presented here could then be applied to optimizing the arrangement of the groups and structures identified or produced by such a system.

References

[1] M.-J. Kraak and F.J. Ormeling, *Cartography*: *Visualization of Spatial Data*, Longman, Harlow, 1996.

[2] C.B. Jones, *Geographical Information Systems and Computer Cartography*, Longman, Harlow, 1997.

[3] R. Weibel and G.H. Dutton, Constraint-based Automated Map Generalization, in *Proceedings of the Eighth International Symposium on Spatial Data Handling*, International Geographical Union, 1998, pp. 214–224.

[4] B.G. Nickerson and H. Freeman, Development of a rule-based system for automatic map generalization, in *Proceedings of the Second International Symposium on Spatial Data Handling*, International Geographical Union, 1986, pp. 537–556.

[5] C.B. Jones, G. Ll. Bundy and J.M. Ware, Map Generalization with a Triangulated Data Structure, *Cartography and Geographical Information Systems*, 22 (1995), 317–331.
[6] J.M. Ware and C.B. Jones, Conflict Reduction in Map generalization Using Iterative Improvement, *GeoInformatica*, 2 (1999), 383–407.
[7] L.E. Harrie, The Constraint Method for Solving Spatial Conflicts in Cartographic Generalization, *Cartography and Geographic Information Science*, 26 (1999), 55–69.
[8] A. Ruas, A method for building displacement in automated map generalization, *International Journal of Geographical Information Science*, 12 (1998), 879–803.
[9] P. Hojholt, Solving Local and Global Space Conflicts in Map Generalization Using a Finite Element Method Adapted from Structural Mechanics, in *Proceedings of the Eighth International Symposium on Spatial Data Handling*, International Geographical Union, 1998, pp. 679–689.
[10] W.A. Mackaness and R. Purves, Automated Displacement for Large Numbers of Map Objects, *Algorithmica* (this issue).
[11] J. Christensen, J. Marks and S. Shieber, An Empirical Study of Algorithms for Point-Feature Label Placement, *ACM Transactions on Graphics*, 14 (1995), 203–232.
[12] S.A. Hirsch, An Algorithm for Automatic Name Placement Around Point Data, *The American Cartographer*, 9 (1982), 5–17.
[13] T. Dickerson and R.S. Drysdale, Fixed-Radius Near Neighbours Search Algorithms for Points and Segments, *Information Processing Letters*, 35 (1990), 269–273.
[14] W.A. Mackaness and M.K. Beard, Use of Graph Theory to Support Map Generalization, *Cartography and Geographic Information Systems*, 20 (1993), 210–221.
[15] N. Regnauld, Recognition of Building Clusters for Generalization, *Advances in GIS Research II*, Taylor and Francis, London, 1996, pp. 185–198.