Resolving Graphic Conflict in Scale Reduced Maps: Refining the Simulated Annealing Technique

Mark Ware[1], Nathan Thomas[1] and Christopher Jones[2]
[1]School of Computing
University of Glamorgan
Pontypridd CF37 1DL
Wales, UK
jmware@glam.ac.uk

[2]Department of Computer Science
Cardiff University
Cardiff CF24 3XF
Wales, UK

Introduction

In previous work, the authors show the potential for iterative improvement techniques to be used as part of an automated map generalisation solution ([1], [2]). In particular, they present a simulated annealing algorithm that controls operations of displacement, deletion, reduction and enlargement of multiple map objects in order to resolve graphic conflict arising as a consequence of scale reduction. The algorithm adopts a trial position approach in which each of n discrete polygonal objects is assigned k candidate trial positions that represent the original, displaced, deleted, reduced and enlarged states of the object. This gives rise to a possible k^n distinct map configurations; the expectation is that some of these configurations will contain reduced levels of conflict. Each configuration has an associated overall cost, which can be computed. This overall cost combines both conflict cost (i.e. the extent to which acceptable minimum clearances between map objects are violated) and modification cost (i.e. the extent to which the map has been altered). Finding the configuration with least overall cost by means of an exhaustive search is not practical for realistic values of n and k. However, it has been shown that near optimum solutions can be found by using simulated annealing to direct a search through a subset of the configurations; thus effective resolution of graphical conflict can be achieved.

High-Order Feature Alignment

A shortcoming of the existing algorithm is that conflict cost depends only on the extent to which minimum clearance constraints are violated, and modification cost is calculated simply as a function of the degree to which each generalisation operator is applied. This approach can be regarded as primitive in that more subjective elements of map quality are overlooked. The specific problem that we address here is that of maintaining building alignment. One way in which a map can suffer deterioration in quality during generalisation is if a meaningful grouping of objects (or *high-order feature*) looses its shape to such an extent that the group is no longer recognisable. For example, displacing a particular building object might result in a row of building objects, representing a street, becoming misaligned or fragmented, thus rendering the street unrecognisable (Figure 1). Experiments show this type of problem to occur in practice.

Two possible solutions are being considered currently, both of which are conceptually straightforward. The first involves assigning a relatively high cost to situations where misalignment occurs. The hope is that such a strategy would both discourage offending displacements from taking place in the first place, and also stimulate remedial courses of action if and when misalignments do appear (e.g. realign buildings by performing additional displacements). The second solution involves high level feature modification (e.g. if a particular object is displaced, and that object forms part of a high level feature, then all other objects belonging to the same high level feature undergo the same displacement). There are possibly some added advantages with this approach in that the search space is made smaller (meaning the algorithm will run faster), modification operators can be applied in a more consistent fashion, and there is the possibility that remote solutions will be found more easily. A disadvantage is that conflict between objects belonging to the same high level feature will not be resolved (although post-processing of the data using solution 1 could be used to overcome this).

High-order features, such as a row of buildings representing a street, are not explicitly defined within source datasets (as is the case with OS MasterMap data). The first problem to solve is therefore that of banding together low-order features, such as an individual building, into higher-order features, such as streets. Finding a solution to the automated grouping of objects is a non-trivial task; previous work of note in this area has been carried out by Regnauld [3] and Christophe and Ruas [4]. We intend adopting an approach in which building objects are grouped on the basis of a range of variables, including: minimum separating distance between buildings, initial alignment of buildings, relative orientation of buildings, proximity to roads, shape and size of buildings, and building attribute information.

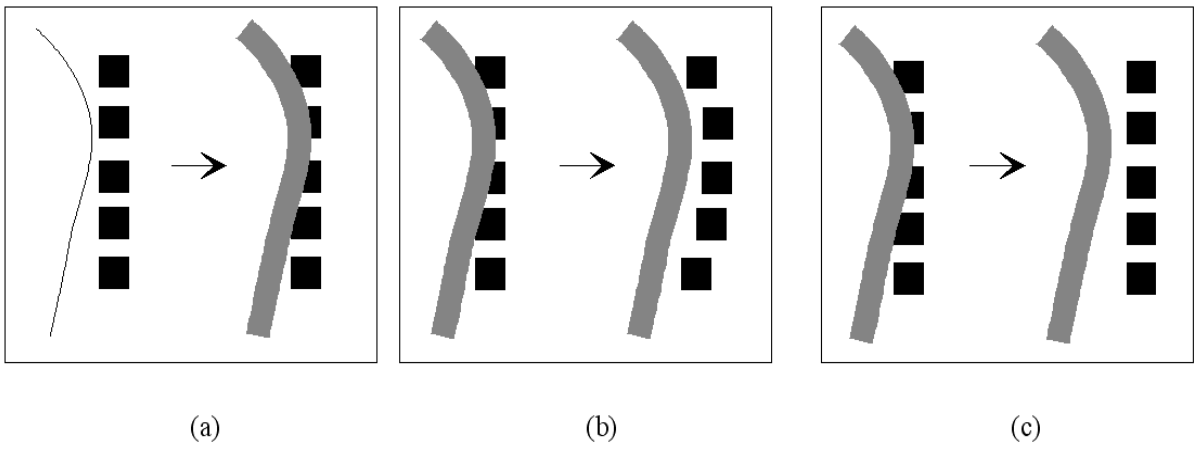
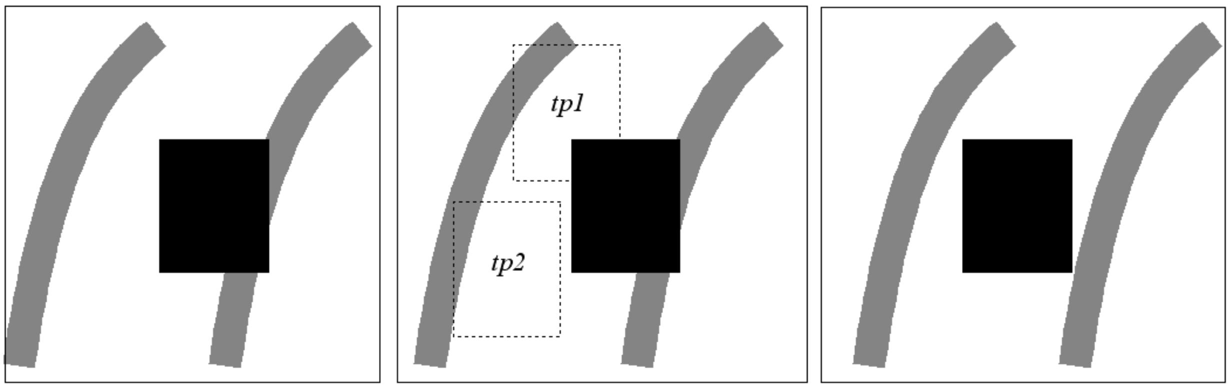


Figure 1. (a) Road symbolisation results in overlap with buildings. (b) Displacement of individual buildings resolves overlap but leads to misalignment. (c) Alignment maintained by treating row of buildings as a group.

Continuous Search Space

The use of fixed trial positions for the displacement of areal objects for the purpose of graphical conflict resolution has been shown to be successful. However in some circumstances, the use of a discrete search space can lead to problems. For example, consider Figure 2, in which the building feature is in conflict with a symbolised road feature.

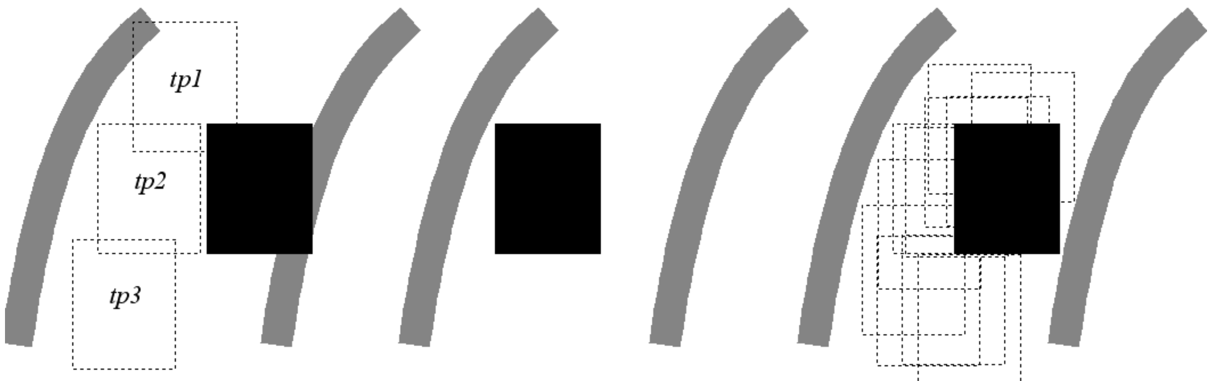


(a) (b) (c)

Figure 2. (a) Enough space exists for the object to move to in order to resolve spatial conflict. (b) However, none of the available trial positions resolve conflict. (c) The desired solution.

It can be seen that displacing the building slightly to the left would resolve all graphical conflict. Unfortunately, a trial position corresponding to such a displacement does not exist; in fact, displacing the building to any of its available trial positions results in further conflict. A similar, if perhaps less critical, problem is illustrated in Figure 3. As before, graphical conflict can be resolved by displacing the building feature slightly to the left, but, again, a trial position corresponding to such a displacement does not exist. This time graphical conflict can be resolved by making use of one of the available trial positions; however, the problem we are faced with now is that the displacement appears excessive. The same problem can occur when applying the reduction operator.

A possible solution is to increase the resolution of the search space by adding to the number of displaced and reduced trial positions. This approach will be investigated, but there are two obvious limitations. First, it is clear that however many trial positions are added, the search space remains discrete and there is no guarantee that in any given situation the appropriate displacement or reduction will exist as a trial position. Second, increasing the number of trial positions increases the size of the search space and this is likely to have a negative impact on execution times.



(a) (b) (c)

Figure 3. (a) Available trial positions. (b) Conflict is resolved, but displacement is excessive. (c) Increasing resolution of trial solutions might provide the required solution.

An alternative solution might be to adopt a continuous search space, as advocated by Strijk and van Kreveld [5] for the purpose of point labelling. The idea we have is quite simple and involves the generation of random trial positions on the fly. The simulated annealing algorithm will work in very much the same way as before, with its main loop again beginning by choosing a modifiable object at random. However, instead of then choosing a trial position at random, the next step will involve the selection of an operator at random (i.e. displace, reduce, enlarge or delete) together with appropriate randomly generated operator parameters (i.e. if displace is selected then a random displacement vector is generated, and if reduce is selected then a random reduction factor is generated). The operator, and parameters, is then applied and the algorithm proceeds as before. By adopting this approach we will hopefully overcome the problem of having only a limited solution space. The hope is that the increased number of alternative realisations will produce improved results (i.e. minimum graphical conflict with least amount of modification).

Initial Results

Some of the ideas presented in previous sections have been implemented and tested using building polygon data extracted from OS MasterMap data and road centre lines taken from OS Oscar data. A simple building grouping function has been developed that groups building features together on the basis of both proximity and size. Each building feature is assigned to one group only. Output resulting from the application of this function is shown in Figure 4.

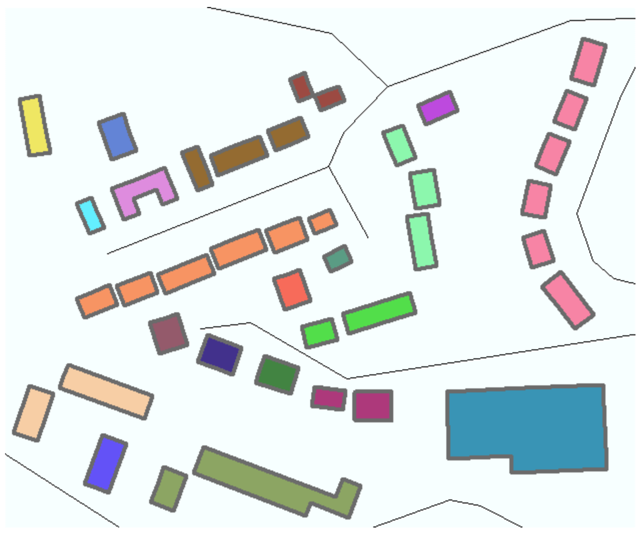


Figure 4. Building features, coloured according to group.

The simulated annealing algorithm described previously has been adapted so that map modifications are applied to groups as opposed to individual buildings. Consider a group g_1, consisting of three buildings b_1, b_2 and b_3. If g_1 is chosen to be modified (e.g. displaced) then the displacement is applied to b_1, b_2 and b_3. Figure 5 shows two examples of output produced with and without grouping.

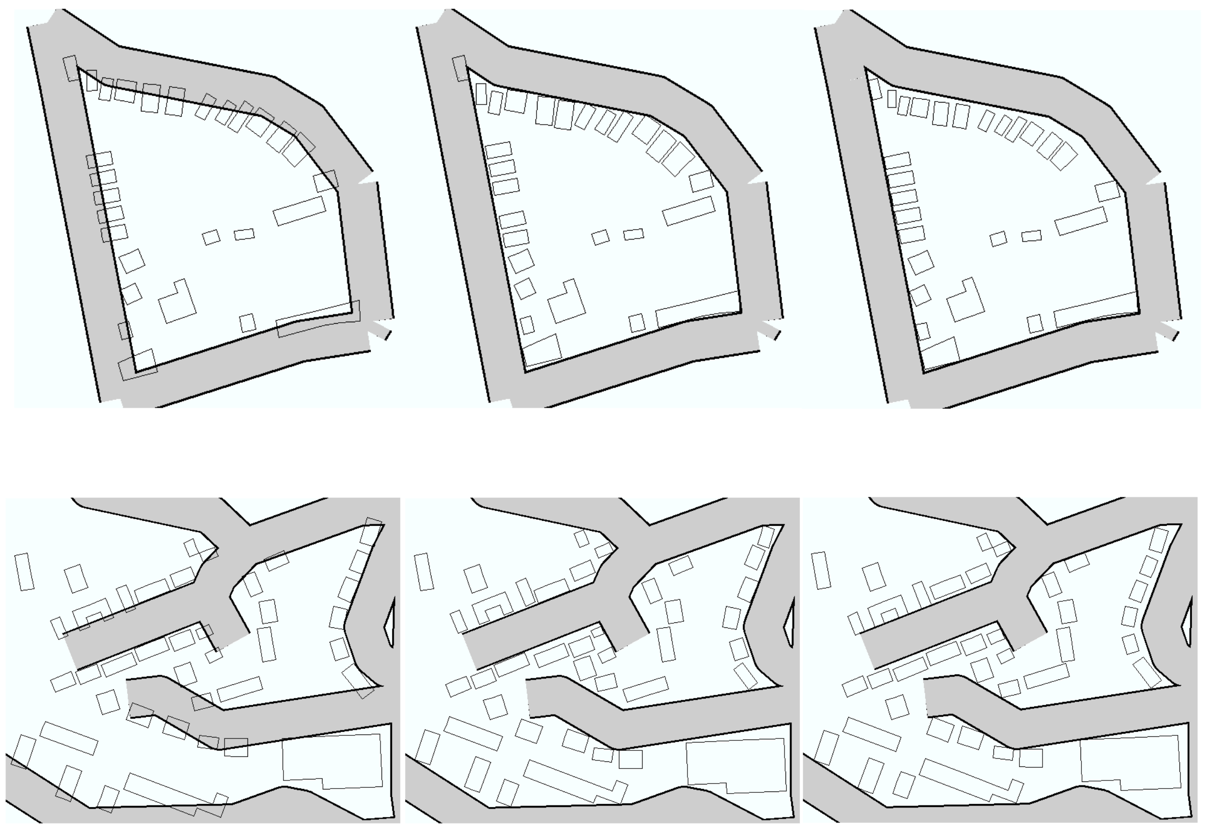


Figure 5. Symbolising road features results in graphic conflict (left). Application of simulated annealing (SA) to individual building features can lead to misalignment and fragmentation (middle). Application of SA to groups maintains high-order features (right).

The fixed trial position approach has also been replaced. Instead, modification operator parameters are generated randomly as part of the algorithm. This, in effect produces a continuous solution space. Figure 6 give example output produced using the alternative approaches.

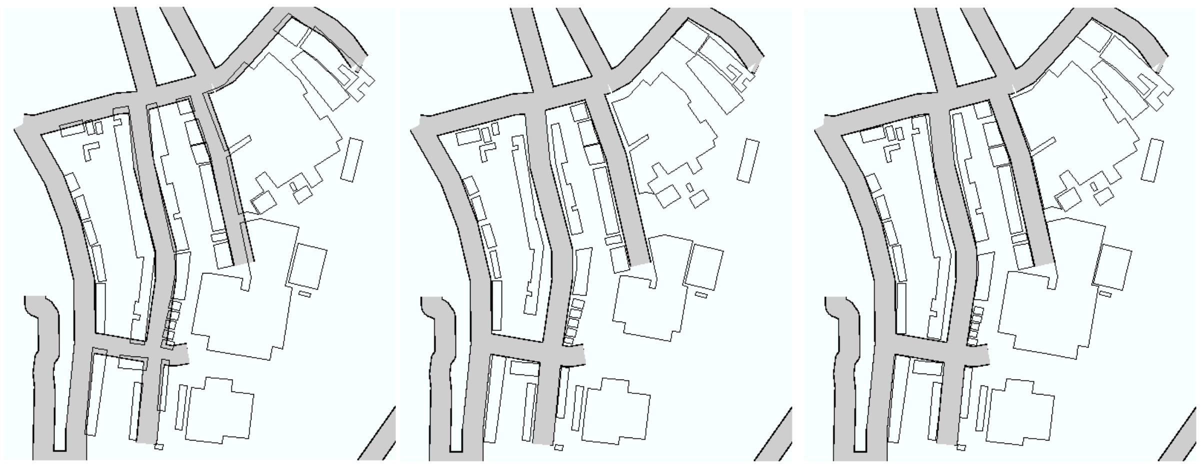


Figure 6. Symbolising road features results in graphic conflict (left). Application of SA using fixed trial positions results in excessive displacement and excessive reduction of building features (middle). Application of SA using randomly generated trial positions reduces displacement and reduction of building features (right).

Repeatable Results

The simulated annealing implementation makes use of the *ran1* function (described in [6]) to generate random numbers. The function is initialised with some arbitrary seed value. Each initialising value will typically produce a different random sequence, and solutions will vary. However, the same initialising value will always produce the same random sequence. This property provides a mechanism for reproducing previous solutions (which is achieved by simply keeping a record of seed values used).

Acknowledgement

This work is partially supported by EPSRC Case Award 00802722, in collaboration with the Ordnance Survey.

References

[1] Ware, J.M. and Jones, C.B., 1998, "Conflict Reduction in Map Generalisation Using Iterative Improvement", Geoinformatica, 2:4, 383-407.

[2] Ware, J.M., Jones, C.B. and Thomas, N., 2001, "Map Generalization, Object Displacement and Simulated Annealing: Two Techniques for Execution Time Improvement", Proceedings of GIS Research UK 2001 Conference (GISRUK'01), 36-38.

[3] Regnault, N., 1996, "Recognition of building clusters for generalisation", Proceedings of 7th International Symposium on Spatial Data Handling, 1-13.

[4] Christophe, S. and Ruas, A., 2002. "Detecting building alignments for generalisation purposes", Proceedings of 10th International Symposium on Spatial Data Handling.

[5] Strijk, T. and van Kreveld, M., 2002. "Practical Extensions of Point Labeling in the Slider Model.", GeoInformatica, 6:2, 181-197.

[6] Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P., 1992, Numerical Recipes in C (book), Cambridge University Press.