# A Tabu Search Approach to Automated Map Generalisation

J. Mark Ware, Ian D. Wilson, J. Andrew Ware
School of Computing, University of Glamorgan
Pontypridd CF37 1DL
Wales, UK
+44 (0) 1443 480480
jmware,idwilson,jaware@glam.ac.uk

Christopher B. Jones
Department of Computer Science, Cardiff University
Queens Buildings, Newport Road, Cardiff CF24 3XF
Wales, UK
+44 (0) 2920 874796
c.b.jones@cs.cf.ac.uk

## ABSTRACT

Displaying map data at scales smaller than its source can result in objects that are either too small to be seen or too close to each other to be distinguishable. Furthermore, graphic conflicts become more likely when certain map symbols are no longer a true scale representation of the feature they represent. Map generalisation includes the processes by which such conflicts are resolved. The map generalisation technique presented here is exponential in the problem size and is, as such, combinatorially large (NP-hard). We show how the tabu search metaheuristic was used to resolve spatial conflict between objects after scaling, achieving near optimal solutions within practical time constraints.

## Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search – *heuristic methods.*

## General Terms: Algorithms.

## Keywords

Graphic conflict resolution, displacement, cartography.

## 1. INTRODUCTION

The task of map generalisation, traditionally the domain of cartographers, is one of selecting and adjusting the symbols on a map to suit the purpose of the map and the scale of the required output [12]. The now widespread use of geographical information systems (GIS), with their integral capacity for producing maps, has introduced a requirement to include a facility for map generalisation within these systems. Map production systems use data that is in some sense pre-generalised, having in many cases been derived from the cartographic products that are based on a particular scale of representation. However, if a map is displayed at a significantly reduced scale, data objects that were clearly visible as separate entities can become too small to be seen or too close to each other to be distinguishable. In particular, these graphic conflicts arise when the map symbols are no longer a true scale representation of the feature they represent. For example, a road symbol may be much wider, when map scale is taken into account, than the width of the road on the ground (Figure 1).

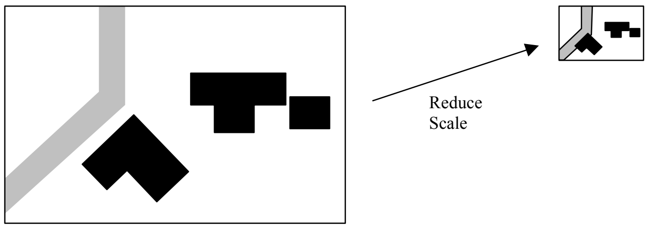

**Figure 1. Example of conflict between objects displayed at a reduced scale.**

in automated cartographic design has increased significantly over the past decade. For further information the reader is directed to collections of papers dedicated to automated map generalisation ([2], [11], [17], [18]). Automation of individual operators required to perform generalisation has been the focus of much of the work to date. These operators are summarised as follows [13]:

- Reduction of the detail in linear features and boundaries;
- Elimination of features that may be too small to discern;
- Collapse in the dimensionality of areal features to lines or points;
- Amalgamation of adjacent features of the same or similar category;
- Exaggeration of important features otherwise too small to represent;
- Typification (or caricature) of the form of features as part of the process of detail reduction;
- Displacement of adjacent features that are in graphic conflict with each other.

Automation of some of the individual generalisation operators can be found in commercial GIS systems such as ESRI's ArcGIS, Intergraph's Map Generalizer and Laser-Scan's Lamps2. Use of the operators however still requires manual process control that involves the user in deciding which operators to apply, in which order, and how they should be utilised in terms of relevant distance tolerances and other control parameters. Brassel and Weibel state that automating process control is essential if map generalisation is to graduate from an interactive user-controlled procedure to one that is fully automated [1].

Effective map generalisation involves careful examination of the interactions between all map symbols. These interactions may give rise to obvious graphic conflicts of proximity and overlap. They may also determine whether important messages, regarding the structure and form of the mapped features, are effectively communicated (for example, in the alignment of buildings, parallelism between neighbouring rivers and roads, and the clustering of woods and lakes). Graphic conflict can be addressed by a combination of possible actions such as elimination, displacement, amalgamation and boundary simplification, combined with appropriate techniques for evaluating the quality of the result. However, the application of an individual operator may have an effect on a map symbol that was not previously in conflict, resulting in propagation of conflict within the map space. A partial solution to this problem is for objects to be moved after scaling so that they remain distinct, visual entities.

## 2. PROBLEM SIZE AND COMPLEXITY

The research presented here describes a procedure that makes use of displacement of multiple map objects in order to resolve graphic conflict. The procedure incorporates a trial position approach similar to that used in [19], and compliments the work presented in [15] (which differs in that it made use of simulated annealing, as opposed to tabu search). Here, each of *n* discrete polygonal objects is assigned *k* candidate trial-positions, which represent displaced states of the object, into which they can possibly move. This results in a possible $k^n$ distinct map configuration; the assumption being that some of these configurations will contain less conflict than the original. Finding an acceptable configuration by means of an exhaustive search is, however, not practical for realistic values of *n* and *k*. The map generalisation solution presented here is therefore a combinatorial problem, the size of which depends upon the number of objects represented and the position of each object within an (x, y) co-ordinate space. Given that it is undesirable to move an object too far from its original position, it is necessary to constrain its movement to within a short distance, which reduces the co-ordinate space considered.

## 3. TABU SEARCH

Tabu search ([4], [5]) is a procedure for solving discrete combinatorial optimisation problems. Glover provides evidence of both the adaptability and efficiency of the approach, with it having been successfully applied to obtain optimal or near optimal solutions to such problems as scheduling, timetabling, travelling salesman and layout optimisation [6]. Recently the technique has been applied successfully to the problem of point-feature cartographic label placement [14]. Tabu search can be viewed as an iterative technique that explores a set of feasible states by a sequence of moves. In general, the best move is taken at each iteration. However, to help prevent the search process from returning a local optimum, a move may be accepted that is as good as or worse than that of the current solution. To avoid repeatedly returning to a previous solution (called cycling), some moves, at each iteration, are classified as illegal, or tabu. Tabu moves are based on the short-term and long-term history of the sequence of moves that meet defined criteria. For example, one might classify a move as tabu if the reverse move has been made recently (within a given number of iterations) or frequently (a given number of times). It may sometimes be desirable to make an otherwise tabu move; in order to achieve this, a particular implementation may include aspiration criteria that override the tabu status of particular moves. Such aspiration criteria might include a case which, by ignoring that a move is tabu, leads to a solution that is the best obtained so far.

The search space $S$ to which tabu search can be applied can be characterised as a set of *i* moves $M=\{m_1,...,m_i\}$ and the application of the moves to a feasible solution $s \in S$ that leads to *i* usually distinct solutions $M(s)=\{m_1(s),..., m_i(s)\}$. The subset $N(s) \subseteq M(s)$ of feasible solutions is known as the neighbourhood of *s*. The method commences with a solution $s_0 \in S$ and determines a sequence of solutions $s_1, s_2,...,s_i \in S$ by a succession of *i* moves. At each iteration a solution is selected from the neighbourhood ($s_{i+1} \in N(s_i)$). The selection process first determines the current solution's $s_i$ tabu set of neighbours $T(s_i) \subseteq N(s_i)$ and the aspirant set of tabu neighbours $A(s_i) \subseteq T(s_i)$. Then the new solution is the best aspirant or non-tabu neighbour. The tabu search procedure is halted when a given threshold for an acceptable solution has been achieved or when a certain number of iterations have been completed.

## 4. PROBLEM DECOMPOSITION

In this section, considerations relating to the map generalisation model are described, and an overview of the model's underlying representation and physical implementation is provided, along with a detailed discussion about the objective function and its mathematical formulation.

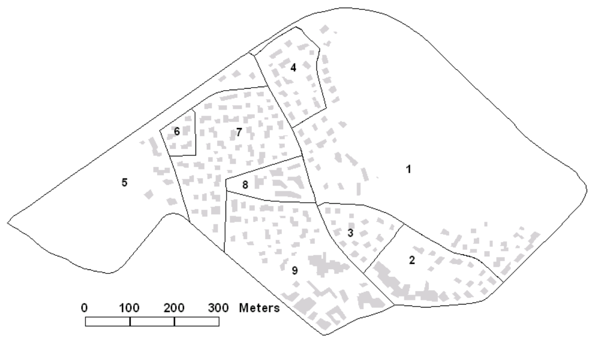

**Figure 2. Division of BDTopo data into 9 autonomous segments.**

### 4.1 Map Segmentation

Determining the near optimum allocation of objects with the co-ordinate space of even relatively small maps is too computationally explosive to be considered for use in most applications. However, the size of the search space, and hence the number of configurations having to be generated and evaluated, can be reduced by dividing the map into autonomous regions, or segments. These segments contain a set of objects such that there is no possibility of these objects coming into conflict with objects in any other segment. In some instances it may be possible to make use of naturally occurring segments, such as those formed by a road network or administrative boundaries. Other situations will require analysis of the distribution of objects in order to find groupings of objects that sustain no influence from objects external to their group. The work presented here makes use of a road network to divide the map into segments (illustrated in Figure 2).

### 4.2 Underlying Structure

A map display is made up of fixed linear objects and modifiable detached polygonal objects. Available search space is the whole x, y co-ordinate space, but this is unrealistic and unnecessary; instead a discrete neighbourhood of permissible moves (trial positions, illustrated in Figure 3) is associated with each modifiable object. The final trial position an object can occupy can not extend beyond a given distance, *d*, from its starting point, $tp_1$. Each modifiable object $o_i$ has $k$ possible states ($tp_1, tp_2, \ldots tp_k$), providing a total of $k^n$ possible map configurations. At any given time, an object exists in one of its trial positions ($tp_{o_i c}$). An object's initial map position is designated as being trial position one.

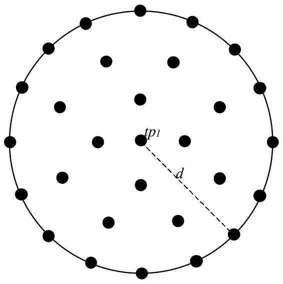

**Figure 3. Example of possible trial positions associated with a modifiable object (*$tp_1$* represents the object's starting position).**

## 4.3 Map Display Evaluation

The success of any discrete optimisation problem rests upon its cost function, the purpose of which is to provide a measure for any given solution that represents its relative quality. The cost function used here works by calculating and summing the costs associated with the collection of objects within our state representation. When first invoked the fitness function calculates the cost associated with every object. A record of these costs is maintained for future reference, meaning that, only the cost connected with objects affected by the most recent displacement are evaluated.

The cost associated with a given configuration is an abstraction of the fitness of the relationship between each conflicting polygonal and linear object represented. A spatial index together with a search procedure (detailed in [15], [8], [9]) is used to identify conflicting objects quickly. The extent to which an object is in conflict determines its individual associated cost. We consider two categories of spatial conflict:

- Conflict between a pair of polygonal objects, where their proximity renders them indistinguishable from each other). This conflict occurs when the minimum separating distance (in viewing co-ordinates) between two objects is less than some redefined threshold.
- Conflict between a polygonal object and a linear object, where their proximity renders them indistinguishable or they overlap. This conflict occurs when the minimum separating distance (in viewing co-ordinates) between a polygonal and linear object is less than some redefined threshold.

### *4.3.1 Underlying Model and Definitions*

The object function used to evaluate solutions to the map generalisation problem requires a number of definitions that model the problem's underlying structure, specifically:

- $O$: $\{o_1, \ldots, o_n\}$ is the set of all polygonal objects;
- $L$: $\{l_1, \ldots, l_r\}$ is the set of all linear objects;
- $n$ is the number of polygonal objects;
- $r$ is the number of linear objects;
- $do_{min}$ is the minimum distance threshold between polygonal objects;
- $dl_{min}$ is the minimum distance threshold between linear and polygonal objects;
- $DO_{ij} = 1$ if $o_i$ is within $do_{min}$ of $o_j$ else 0;
- $DL_{ij} = 1$ if $o_i$ is within $dl_{min}$ of $l_j$ else 0.

### *4.3.2 Object Relationship Fitness Function*

The objective function used to evaluate solutions to the map generalisation problem examines the weighted relationship between linear and polygonal objects. The general expression of the objective function is:

$$f = ((f_1 * w_1) + (f_2 * w_2)) \qquad (1)$$

Here $f_i$ and $w_i$ represent, respectively, the number of conflicting objects and the weight of that particular measure, with a low value of $f$ indicating a good solution.

The first term of the objective function, $f_1$, counts the number of polygonal objects that conflict with each polygonal object. Minimising the number of polygonal objects in conflict with each other produces a more attractive map display.

$$f_1 = \sum_{i=1,n} \sum_{j=1,n} DO_{ij} \qquad (2)$$

The second term of the objective function, $f_2$, counts the number of polygonal objects that conflict with each linear object. Again, minimising the number of polygonal objects in conflict with each linear object produces a more attractive display.

$$f_2 = \sum_{i=1,n} \sum_{j=1,n} DL_{ij} \qquad (3)$$

## 5. TABU SEARCH SOLUTION

In this section, we describe our implementation of the tabu search algorithm for solving the map generalisation problem. The notation used follows that presented in [7].

## 5.1 Components

### *5.1.1 Objective Function*

For a map display containing $n$ modifiable polygonal and $r$ fixed linear objects a configuration $s$ corresponds to an individual arrangement of these objects. For each solution $s \in S$, $f(s)$ is a measure of the total number of polygonal and linear objects in conflict (see equation (1) shown in section 4.3.2).

### *5.1.2 Tabu List*

Our recency list of tabu moves $T_m$ was implemented as a dynamically-sized stack and is comprised of a set of <*object, trial-position*> pairs. The size of $T_m$ is set proportional to the amount of conflict (as suggested in [14]) and is given an initial value ($c_1 + c_2$*f). The size is recalculated every $c_3$ consecutive iterations. Suitable values for $c_1$, $c_2$ and $c_3$ are arrived at empirically. When an object is moved to a trial position, the new <*object, trial-position*> pair is pushed onto $T_m$. When $T_m$ is full, the oldest move is removed. Unless aspiration criteria are met, trial position $tp_j$ is unavailable to object $o_i$ for as long as $<o_i, tp_j>$ remains in $T_m$. In addition to our recency list, we implement an unbounded tabu list, $T_s$, of historical solutions, which prevents the long-term cycling of solutions. At each iteration the solution $s = \{\{o_1, tp_{o_1 c}\} \ldots, \{o_n, tp_{o_n c}\}\}$ is added to the historical list $T_s$; each new trial solution is compared to the elements of this list, preventing solutions previously expanded upon from being revisited.

### 5.1.3 Candidate List

Solutions $s$ and $s' \varepsilon S$ are neighbours if they are different at a single iteration, providing a neighbourhood function $N(s)$. A neighbour of $s$ can be obtained by changing the location of an object $o_i$ in $s$ in such a way that the new location satisfies the trial position constraint. Given this, a move is identified by a triplet $<o, tp, f(s')>$, being respectively an object, a trial position for that object and objective score for the solution generated. All such triplets are stored in a candidate list $V^*$, which is sorted by $f(s')$.

### 5.1.4 Aspitation Criteria

An aspiration criterion similar to that suggested in [7] was implemented. Specifically, a move that would otherwise be considered tabu can be made if it leads immediately to a solution that is better than the best solution $s^*$ encountered so far. In addition, if all candidate moves are tabu or if all neighbouring solutions are tabu, then the move with higher permanency time in $T_m$ is applied (this situation does not arise in practice).

### 5.1.5 Incremental Evaluation

The algorithm makes use of and incremental evaluation method similar to that proposed in [3]. Our objective function $f(s)$ measures the extent to which map objects are in conflict. The cost $f(s^{init})$ of an initial map configuration is found by assigning the correct value to all $DO_{ij}$ and $DL_{kl}$ pairings, before applying equations (1), (2) and (3). The neighbours of $s^{init}$ are obtained by changing the location of objects $o$ in $s^{init}$ in such a way that the new location satisfies the trial position constraint. Given this, the move required to generate a particular neighbour of $s^{init}$ is identified by a triplet $<o_i, tp_j, f(s')>$, being respectively an object, a trial position for that object and objective score for the solution generated. Calculation of $f(s')$ requires a re-evaluation of the conflict associated with $o_i$ only. In our implementation, the list $V^*$ of all possible moves from $s^{init}$ is recorded. The successor $s$ to $s^{init}$ is found by application of a least-cost valid move (this move must be non-tabu or aspirant and $s$ must be non-tabu). Given that object $o_m$ is moved, $V^*$ is maintained by updating the cost associated with any moves involving $o_m$ and any moves involving objects currently or immediately previously in conflict with $o_m$. All subsequent configurations are arrived at by applying a least-cost valid move taken from $V^*$, which is then updated.

### 5.1.6 Stopping Conditions

A key factor governing the success of tabu search is to know when to stop searching. A search should stop if an optimal solution is found. However, finding an optimal solution might take an unacceptable length of time. Furthermore, the only way of knowing for sure if a particular state $s^g$ represents a global optimum is when $f(s^g)=0$; for many problems there is no guarantee that the optimal solution has zero cost. Therefore, there is the need for additional stopping conditions. These can include limits on the number of iterations, elapsed time, memory usage or total improvement in the solution cost [10]. In our algorithm, the search stops if the number of moves made without an overall improvement exceeds a pre-set limit $I$ (or if $f(s)=0$ is found).

## 5.2 Tabu Algorithm

### 5.2.1 Definitions

The algorithm used to search the state-space associated with the map generalisation problem requires a number of variable and function definitions, specifically:

- $s$: the current solution;
- $f^*$: cost of best configuration encountered so far;
- $V^*$: candidate list (each item has form $<o, tp, f(s')>$);
- **IV**$^*(s)$: initialise candidate list based on configuration $s$;
- $s^*$: best solution found;
- **IT**: initialise tabu list;
- $I$: max number of moves allowed without overall improvement;
- **GF**$(V^*)$: return first item in list;
- $m^*$: candidate move $(m^*_o, m^*_{tp}, m^*_f)$;
- **T**: returns TRUE if particular move or state is tabu;
- **GN**$(V^*)$: return next item in list;
- **M**$(o, tp, s)$: move object $o$ to trial position $tp$ in configuration s, return new configuration;
- $ts$: temporary configuration;
- **UT**: insert move or state at head of tabu list.

### 5.2.2 Tabu Search Algorithm

```
TabuSearch(s, I)
f*←f(s); V*←IV*(s); i←0; Tm←IT(); Ts←IT(); s*←s
WHILE i≤I AND f*>0 DO
    m*←GF(V*)
    WHILE T((m*o, m*tp),Tm) OR T(ts←(m*o, m*tp, s), Ts) OR m*≠NULL DO
        m*←GN(V*)
    END
    IF m*=NULL THEN
        (m*o, m*tp) ←GF(Tm)
    END
    s←M((m*o, m*tp), s); f'←m*f; Tm←UT((m*o, m*tp), Tm); Ts←UT(s, Ts)
    IF (f'<f*) THEN
        s*←s; f*←f'; i←0
    ELSE
        i←i+1
    END
END
RETURN(s*)
```

## 6. COMPUTATIONAL EXPERIMENTS

The algorithm was implemented in C running under UNIX on a Sun Enterprise 2 model 2200 (2x200MHz Ultrasparc). Initial experiments made use of IGN-France BDTopo data (source scale 1:25,000) consisting of 214 polygonal objects contained within 9 segments (Figure 2). Algorithm parameters were configured as follows: the tolerance values $do_{min}$ and $dl_{min}$ were set to 10.0m and 5.0m respectively; maximum displacement value $d$=10m; conflict between pairs of polygons is deemed less costly than conflict involving polygons and linear features and the cost values $w_1$ and $w_2$ are set so as to reflect this fact (10 and 1 respectively); $c_1$=50, $c_2$=5 and $c_3$=300; in experiment1 $I=10n$, in experiment2 $I=100n$; our abstraction of the displacement space $d$ provided twenty-nine trial positions for each object. Results are presented in Table 1.

**Table 1. Results (TS - tabu search, GD - gradient descent)**

| SEGMENT | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Objects *n*** | | 43 | 16 | 17 | 21 | 9 | 6 | 55 | 6 | 41 |
| $f(s^{init})$ | | 78 | 76 | 50 | 52 | 30 | 30 | 106 | 24 | 92 |
| ***TS*** | $f^*$ | 10 | 10 | 0 | 2 | 0 | 2 | 0 | 2 | 12 |
| ***I=10n*** | **Time (s)** | 1.82 | 0.64 | 0.06 | 1.2 | 0.02 | 0.44 | 1.41 | 0.58 | 2.17 |
| ***TS*** | $f^*$ | 2 | 10 | 0 | 0 | 0 | 2 | 0 | 2 | 2 |
| ***I=100n*** | **Time (s)** | 18.9 | 23.8 | 0.06 | 1.86 | 0.02 | 5.33 | 1.44 | 5.87 | 26.1 |
| ***GD*** | $f^*$ | 30 | 12 | 0 | 10 | 0 | 2 | 16 | 2 | 12 |
| | **Time (s)** | 0.54 | 0.17 | 0.16 | 0.25 | 0.04 | 0.04 | 2.00 | 0.05 | 0.97 |

Experimental data clearly demonstrates how local optimality is escaped (illustrated in Figure 4).

In many cases, solutions known to be optimal (i.e. $f^*=0$) being reached. In all other cases, greatly reduced levels of conflict were achieved.

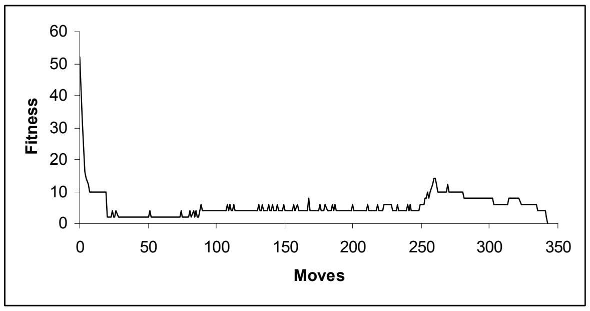

**Figure 4. Escaping local optimality (results from segment 4).**

The tabu search algorithm succeeded in limiting the number of map realisations needed to be generated and evaluated, with processing times ranging between 0.02s and 26.12s. Note that increasing the value $I$ has, in some instances, reduced $f^*$ at the expense of an increase in execution time (e.g. segment 9). In other cases increasing $I$ has had no effect on $f^*$ but has still resulted in increasing execution times (e.g. segment 6). Striking the right balance between quality of result and overall speed of execution would depend to some extent on the intended use of the derived map.

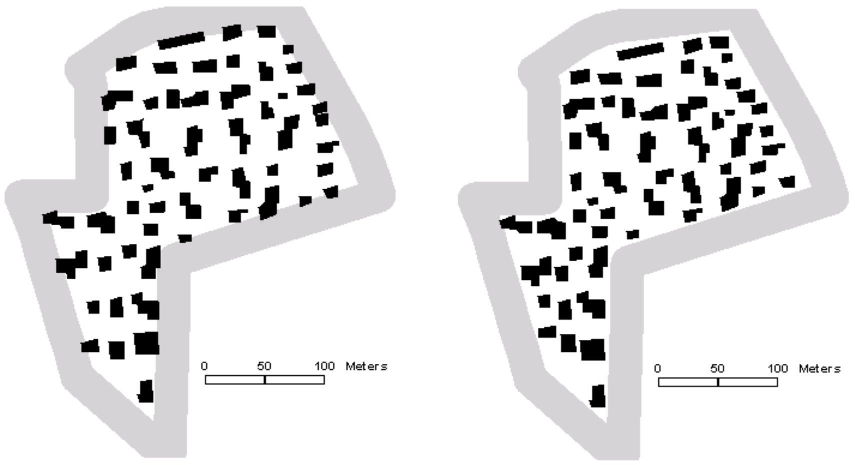

**Figure 5. Object conflict before and after resolution (BDTopo).**

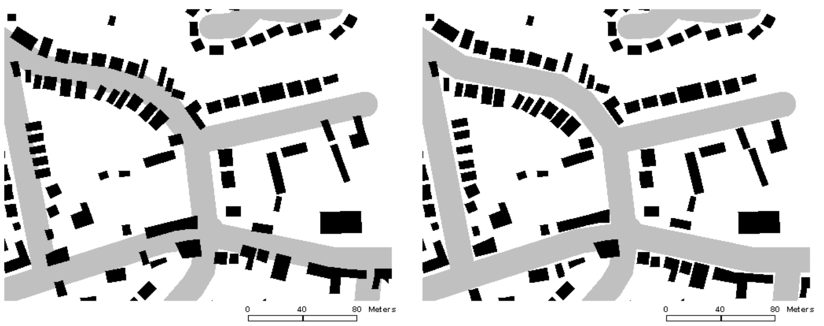

**Figure 6. Object conflict before and after resolution (Mastermap and OSCAR).**

The differences between the starting and final configurations are highlighted in Figure 5 (segment 7). The segment boundary has been symbolised and is drawn with width 10.0m. Experiments to compare tabu search results with a simple gradient descent (GD) implementations were also carried out, with results also included in Table 1. As is to be expected, GD, with no capacity to escape local optimum, does not perform well in most cases; the main reason for including the results is to demonstrate that our problem space is non-trivial.

The tabu search algorithm has also been applied to Ordnance Survey Mastermap polygon data (building outlines) and OSCAR road centre line data. Sample output is given in Figure 6; road centre lines are drawn with width 20.0m and minimum separation between buildings is set to 2.0m. Again, the algorithm has been successful in resolving the majority of conflict situations.

## 7. CONCLUSION

Given these promising, initial, experiences with the application of tabu search, it is the authors' intention to expand upon the work presented. At present polygons can be displaced from their original location without penalty. In order to minimise the number of displacements of this type, we intend to introduce an additional cost, which will act to encourage polygons to remain near to their original location.

The displacement method presented here works well where there is plenty of free map space into which objects may move. In situations where displacement is either impractical because of space constraints or too expensive in terms of overall disruption to the map display, additional operators (e.g. deletion, amalgamation, reduction and simplification) will be used in combination. Future work will concentrate on introducing these operators. In principle, adding deletion, amalgamation, reduction and simplification capabilities to the current algorithm appears possible; the additional modified states of an object will simply be treated as additional trial positions for that object. The cost function will be expanded to take account of each new operator, with appropriate penalties consistent with the map disruption included.

The two main alternatives to tabu search, simulated annealing and genetic algorithms, have been applied to the same problem ([15], [16]). The authors intend carrying out an intensive comparison of the three approaches in order to ascertain the relative merits of each.

## 8. ACKNOWLEDGEMENT

The authors express thanks to the Institut Géographique National and the Ordnance Survey for permission to use their data in parts of the work presented.

## 9. REFERENCES

[1] Brassel, K.E. and R. Weibel. (1988). "A review and conceptual framework of automated map generalisation", *International Journal of Geographical Information Systems* 2(3), 229-244.

[2] Buttenfield, B.P. and R.B. McMaster. (1991). *Map Generalisation: Making Rules for Knowledge Representation.* Longman.

[3] Fleurent, C. and J. A. Ferland (1996), "Genetic and Hybrid Algorithms for Graph Coloring", *Annals of Operations Research*, 63.

[4] Glover, F. (1989). "Tabu Search-Part I", *Operations Research Society of America Journal on Computing* 1(3), 190-206.

[5] Glover, F. (1990). "Tabu Search: A Tutorial", *Interfaces* 20, 74-94.

[6] Glover, F. (1993). "A user's guide to tabu search", *Annals of Operations Research* 41, 3-28.

[7] Hao, J., R. Dorne and P. Galinier. (1998). "Tabu Search for Frequency Assignment in Mobile Radio Networks", *Journal of Heuristics* 4(1), 47-62.

[8] Jones, C.B. and J.M. Ware. 1998. "Proximity relations with triangulated spatial models", *The Computer Journal*, 41(2), 71-83.

[9] Jones, C.B., J.M. Ware and C.D. Eynon. (1999). "Triangulated Spatial Models and Neighbourhood Search: An Experimental Comparison with Quadtrees", *The Visual Computer* 15(5), 235-248.

[10] Morley, G.D. and W.D. Grover. (2001). "Tabu Search Optimization of Optical Ring Transport Networks", *Proceedings of IEEE Globecom2001*.

[11] Muller, J.C., J.P. Lagrange and R. Weibel. (1995). *GIS and Generalisation Methodology and Practice*. Taylor and Francis.

[12] Robinson, A.H., J.L. Morrison, A.J. Muehrcke, S.C. Guptill and A.J. Kimerling. (1995). *Elements of Cartography.* John Wiley.

[13] Shea, K.S. and R. B. McMaster. (1989). "Cartographic generalisation in a digital environment: When and how to generalise", *Proceedings of 9th International Symposium on Computer-Assisted Cartography*, 56-67.

[14] Yamamoto, M., G. Camara and L. Lorena. (2002). "Tabu Search Heuristic for Point-Feature Cartographic Label Placement", *Geoinformatica*, 6(1), 77-90.

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

[16] Ware, J.M., Wilson, I.D. and Ware, J.A., 2003, "A Knowledge-Based Genetic Algorithm Approach to Automating Cartographic Generalisation", to appear in Knowledge-based Systems.

[17] Weibel, R. (1995). *Cartography and GIS*, Special Issue: Automated Map Generalisation, 22(4).

[18] Weibel, R. and C.B. Jones. (1998). *Geoinformatica*, Special Issue: Map Generalization, 2(4).

[19] Zoraster, S. (1997). "Practical results using simulated annealing for point feature label placement", *Cartography and Geographical Information Systems*, 24(4), 228-238.