# Ant Colony Optimization Applied to Map Generalization

Nigel Richards and Mark Ware (jmware@glam.ac.uk)
Faculty of Advanced Technology
University of Glamorgan
Wales, UK

## 1. Introduction

This paper presents details of an on-going PhD project that is studying the application of Ant Colony optimization to map generalization. One of the objectives of the project is to compare the performance of Ant Colony with a number of well known alternative approaches, including Simulated Annealing. The paper begins by introducing the general ideas of Ant Colony optimization. It then presents two map generalization problems, those of river symbolization and network schematization. An Ant Colony solution to each problem is then described. Experimental results are presented, and compared with Simulated Annealing solutions.

## 2. Ant Colony Optimization (ACO)

An ant when searching for a food source will initially wander randomly. Upon finding food it returns to its colony while laying down a volatile chemical trail called a Pheromone (Goss et al, 1989;). Other foraging ants that smell such a pheromone trail are more likely to be influenced to follow the path to the food source than continuing to wander randomly. Probabilistically, the stronger the pheromone trail, the more likely that an ant will follow the path. Pheromone trails are strengthened by each ant following the path. Initially, there may be many different pheromone trails leading to a single food source. However, over time the strength of the pheromone chemical evaporates reducing its attractiveness to other foraging ants. Thus, as an ant walks, the pheromone it deposits evaporates behind it after a period of time. Longer trails take more time for each ant to walk along and it follows that there is less pheromone density over the trail as a result. Shorter routes to a food source will not take as long to traverse, which helps to maintain a high degree of pheromone density because any evaporation is compensated for by additional pheromone. The more ants that are attracted to the trail the more pheromone density increases. Pheromone evaporation is a vital component to the ant colony to prevent all routes having equal attractiveness. Over a period of time the ants will converge on the shortest path to the food source. This collective ant behaviour of pheromone laying, sensing and following paths to food source was the original inspiration for ACO algorithms. In a simulated system, evaporation prevents an algorithm from converging to a local optimum.

The ACO meta-heuristic (Dorigo et al, 1996) employs a colony of artificial ants that collaborate to find a good solution to a discrete combinatorial optimisation problem. The colony of artificial ants communicates with each other indirectly through the use of artificial pheromone trails. Artificial ants possess some of the characteristics of their real counterparts as well as additional traits that generally suit the optimisation problem at hand. The ACO process will be described in more detail in the next section in the context of a river symbolization problem.

## 3. River Symbolization

On large-scale maps, river networks are usually represented as a series of connected polygons. At reduced scale these polygons are typically replaced by centrelines. These centrelines are themselves likely to require line simplification. In addition, when displaying centrelines (either on paper or electronically), the question of symbolization needs to be addressed (i.e. what line thicknesses should be applied to the centreline?). This is not a straightforward decision since line thickness along the length of the river centreline will not be uniform, but will rather vary according to some criteria – typically based on river width. This sequence of operations is illustrated in Figure 1.

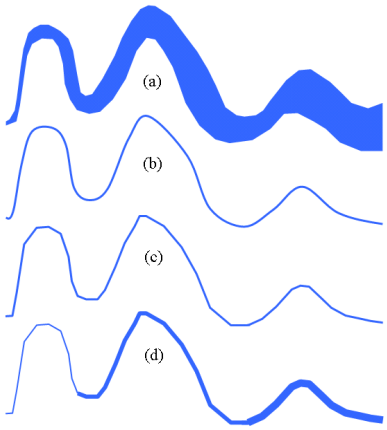



Figure 1 - Generalizing a river polygon. (a) Original polygon. (b) Centreline. (c) Simplified centreline. (d) Symbolized centreline.

The problem addressed here is that of automating the process of river centreline symbolization (Figure 1(d)). A cartographer will usually split the river centreline into a fixed number of segments. The location of each split (and hence the length of each segment) will be decided according to predefined criteria. Each segment is then symbolized (i.e. allocated a line thickness), again in accordance with predefined criteria. Consider the following example criteria:

- River segment width >10 metres (m) shown by single line, 2mm width;
- River segment width between 7m and 10m shown by single line, 1mm width;
- River segment width between 4m and 7m shown by single line, 0.5mm width;
- River segment width < 4m shown by single line, 0.25mm width;

Automating the river segmentation process is non-trivial since, even though the width of a river generally increases when moving from upstream to downstream, this increase will not be monotonic. Therefore, identifying just a single location along a river at which its width equals a particular value is likely to be not possible. Automation therefore requires the use of algorithms that can produce good solutions given the initial predefined criteria, or rules. One approach to developing these algorithms is to consider the problem (i.e. what are the best positions along the centreline at which splits should occur) in terms of a search space, for which there is a set of possible solutions. There are many algorithms that can be used to find an optimal solution; in this paper ACO is used.

*3.1 River Segmentation*

A more formal definition of the problem now follows. Consider a polygon P representing a river, and its associated centreline C, which is made up of n vertices ($v_1$, $v_2$, …., $v_n$). It can be assumed that C has been derived from P using an appropriate triangle-based technique (such as those described by Jones et al (1995) and Regnauld and Mackaness (2006)). The goal here is to divide C into k segments or divisions ($d_1$, $d_2$, …., $d_k$), with divisions having associated average widths W = ($w_1$, $w_2$, …., $w_k$). The average width $w_i$ of a division $d_i$ represents the average width of the corresponding section of P, and is calculated by finding the average height of its constituent triangles. The average widths W are referred to as the target widths, and represent the initial target criteria, or constraints. It makes sense to consider average widths of segments as opposed to actual widths at splitting points because of the non-monotonic way in which a river increases (or decreases) in width when moving in a downstream (or upstream) direction.

Dividing C is not straightforward. For example, consider a simple segmentation that gives rise to $D_A$, consisting of k divisions ($d_{A1}$, $d_{A2}$, …., $d_{Ak}$) of (approximately) equal length, with average widths $W_A$ = ($w_{A1}$, $w_{A2}$, …., $w_{Ak}$). In order for $D_A$ to meet initial criteria, then W and $W_A$ would need to be equal. W and $W_A$ are compared by calculating the value $|W-W_A| = |w_1-w_{A1}| + |w_2-w_{A2}| + \ldots + |w_k-w_{Ak}|$. Typically, W and $W_A$ will not be equal (i.e. $|W-W_A| > 0$). The problem is therefore to automatically reposition the splitting points to produce a segmentation $D_X$ such that $|W-W_X| = 0$. Finding a set of split positions that results in $|w_i-w_{Xi}| = 0$ for all divisions is likely to prove impossible; the problem is therefore redefined as trying to find a set of positions that minimizes width error. The width error $\hat{W}_Y$ is the value $|W-W_Y|$ associated with a segmentation $D_Y$.

Consider the situation where C is divided into 3 divisions of roughly equal length resulting in $D_A$, made up from ($d_{A1}$, $d_{A2}$, $d_{A3}$) where $d_{A1}$ = ($v_1$, $v_2$, …., $v_a$), $d_{A2}$ = ($v_a$, $v_{a+1}$, …., $v_b$) and $d_{A3}$ = ($v_b$, $v_{b+1}$, …., $v_n$). Now consider a slightly changed segmentation $D_B$ made up from ($d_{B1}$, $d_{B2}$, $d_{B3}$) where $d_{B1}$ = ($v_1$, $v_2$, …., $v_{a-1}$), $d_{B2}$ = ($v_{a-1}$, $v_a$, …., $v_b$) and $d_{B3}$ = ($v_b$, $v_{b+1}$, …., $v_n$). That is, $D_B$ is derived from $D_A$ by removing a single vertex (in this case $v_a$) from one segment (in this case $d_{A1}$) and adding to another (in this case $d_{A2}$) resulting in the new segments $d_{B1}$ and $d_{B2}$. In other words, the split location between a pair of adjacent divisions has moved by one vertex in one direction or the other. It is now possible to compare the width errors $\hat{W}_A$ and $\hat{W}_B$ associated with segmentations $D_A$ and $D_B$. Potentially, one of the segmentations will have a smaller error than the other. For example, if $\hat{W}_B < \hat{W}_A$, then it follows that $D_B$ is a better segmentation than $D_A$ (at least, that is, in terms of the target criteria being used). In other words, the reallocation of a vertex has resulted in an improved solution.

If the location of centreline splits is restricted to coincide with vertex locations only, then, based on standard combinatorial theory, the total number of alternative segmentations of C into k segments is:

$$N(S) = (n-2)! / (((n-2)-(k-1))!(k-1)!) \quad \text{- equation (1)}$$

Note that (n-2) is used instead of n since vertices $v_1$ and $v_n$ cannot be considered as splitting locations. Furthermore, it follows that there are k-1 split vertices, each of which joins 2 adjacent divisions. The set of possible solutions S can be regarded as a search space, and the problem of segmentation can be redefined as that of finding the segmentation $D_Z$ in S with smallest width error. One strategy might be to generate and evaluate all possible

combinations. However, this is not feasible for realistic data sets. For example, consider the situation where n=1000 and k=5; this would result in over 40 billion possible segmentations. An alternative strategy for finding $D_Z$ is therefore required.

*3.2 ACO Solution for River Segmentation*

The ACO algorithm is an iterative process. During an iteration each ant attempts to improve the segmentation by moving split positions (by reallocating an edge from one segment to an adjacent segment). Initially, the river centreline to be symbolised is split into the required number of segments (5 in the example shown in Figure 2). The location of initial splits is determined according to predefined criteria.

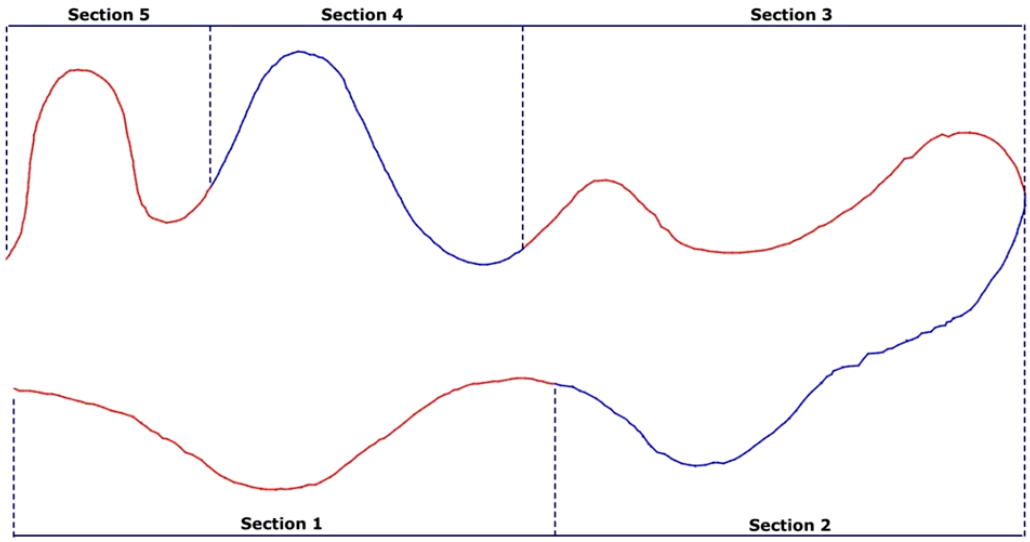



Figure 2 – River centreline divided into 5 initial segments (sections).

The quality of solution produced by ant colony optimisation algorithms is largely dependent on the quality of the associated pheromone trail and heuristic information available to the ants exploring the search space. It is vital therefore, to define a pheromone matrix (Figure 3) that will contain pheromone values that refer to the desirability of moving the boundary of any given segment. In order to achieve this, a pheromone matrix consisting of a pheromone structure for each segment was implemented

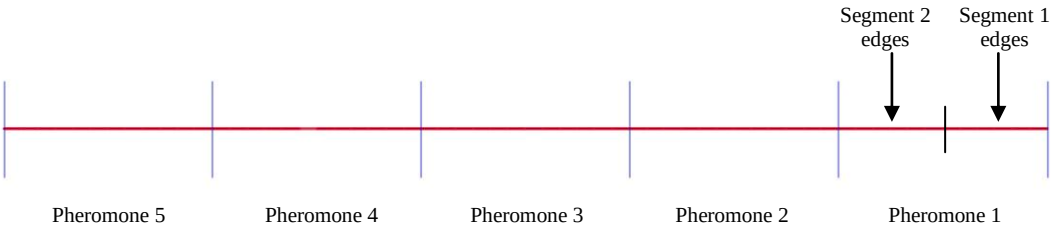



Figure 3 – Pheromone matrix.

Each pheromone structure contains a pheromone value for every edge that may possibly be positioned in its corresponding segment. In the current implementation, edges are allowed to move to immediate neighbouring sections only - so Pheromone 1 (for Section 1) contains

values for all edges in Segment 1 and Segment 2. Pheromone 2 contains values for edges in Segment 1, Segment 2 and Segment 3 – the other Pheromones are constructed on the same basis.

The pheromone value corresponding to each edge in the matrix is initially set to a default value (arrived at via initial experimentation). During an iteration, an ant will consider moving an edge from one segment to another. A State Transition Rule formula is employed to consider the desirability of the move by taking into consideration the pheromone value associated with the edge in question in the pheromone matrix for the section the edge is to be moved into. For example, if an ant is considering moving the boundary edge from segment 1 into segment 2, it will consider the pheromone value for that edge in segment 2 to determine how good such a move has been in the past for other ants. The higher the pheromone value, the more desirable the move is. A low value indicates that the move is not desirable as it has not been beneficial to other ants.

As solutions are constructed, it is important to modify the pheromone matrix in order to influence each ant's decisions whilst searching for solutions. This is achieved through the use of Local and Global pheromone updating rules. The rules ensure that the right balance is maintained between ants exploring the search space and ants exploiting previous knowledge through the pheromone matrix i.e. a good solution has associated high pheromone values in the pheromone matrix.

Figure 4 illustrates a general view of the pheromone matrix after a number of iterations. The red area indicates the strength of the pheromone value for each edge in the matrix.

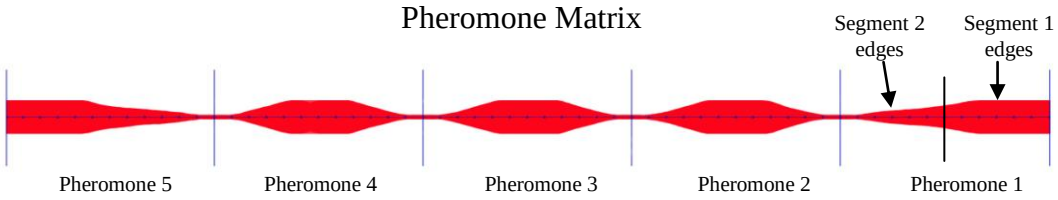


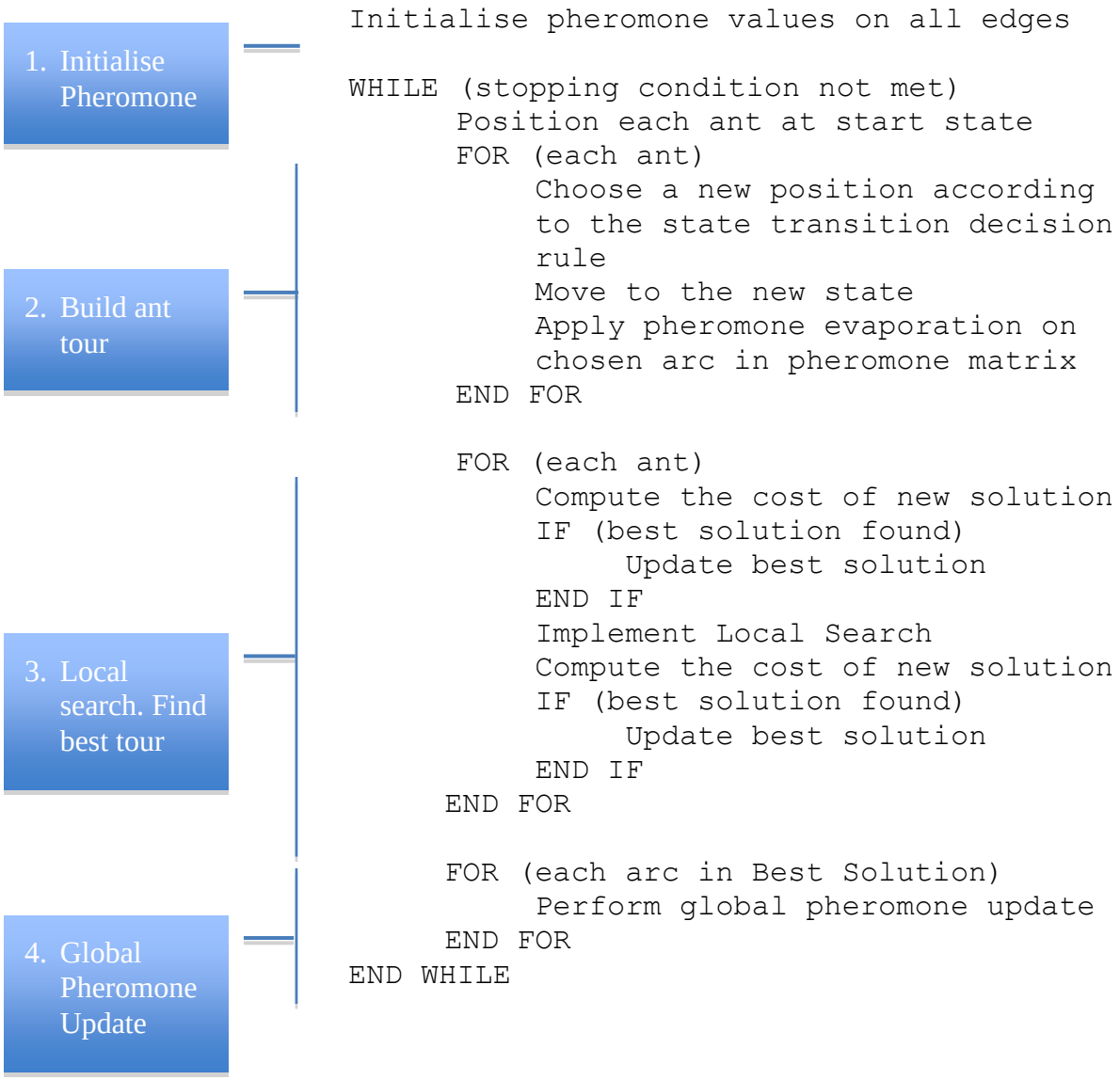

Figure 4 – Pheromone matrix after a number of iterations.

The diagram shows that the pheromone values for edges in segment 1 are strong indicating that better solutions are achieved when they are in segment 1. Segment 2 edges close to the boundary have also resulted in good solutions when located in segment 1, however edges further into segment 2 do not produce such good solutions. Edges at the end of segment 2 have very low pheromone values indicating that it is not desirable to consider moving these edges into segment 1.

In addition to the pheromone matrix, heuristic information is also used in the probabilistic State Transition Rule to determine the ants' next step towards a solution. The ants' can exploit heuristic information in the form of problem-specific knowledge to determine the next course of action during solution construction. In the case of the river symbolisation problem, heuristic information is associated with each segment line and represented by a value corresponding to a fraction of the cost of moving that line from its current segment to an adjacent segment with respect to the current solution.

*3.3 ACO Algorithm*

The overall ACO algorithm can be split into four distinct phases: Initialisation of pheromone, colony tour building and local update, best solution update, and global pheromone update.



Phase 1 requires setting an initial pheromone value for each of the arcs in the problem space. Phase 2 involves the colony of ants constructing their own solutions by using the state transition rule (*pseudo-random proportional rule*) described in the above equation. During each iteration the local pheromone update rule is applied to modify the pheromone value on the arc in question. When the entire colony of ants have completed their tour and found a solution, Phase 3 takes place to record the best solution found to date, followed by Phase 4, which updates the pheromone matrix for the arcs traversed by the ant that found the best solution.

### *3.4 Initial Results*

The ACO algorithm (together with triangulation and centreline generation algorithms) has been implemented using Java. Initial experiments have been carried out using a simple river polygon (Figure 5). Figure 6 shows the corresponding polygon triangulation and river centreline. This centreline is made up from 490 vertices. The goal here is to divide the centreline into 5 divisions ($d_1$, $d_2$, $d_3$, $d_4$, $d_5$), with associated target average widths W = (170.0, 130.0, 90.0, 50.0, 10.0). These widths represent screen units, but could just as easily correspond to appropriate mapping units. Figure 7 shows the initial segmentation (where each segment contains roughly equal number of vertices). This gives an initial width error $\hat{W}_{initial}$ = 87.0. Using equation (1) it follows that the total number of possible segmentations = 2,334,078,990. Figure 8 shows the segmentation after 15000 iterations, at which point the width error has reduced to 19.9 (achieved in a time of less than 1 second). If the optimization is left to run for longer, then the result improves further. The best result achieved to date is a width error of 17.24 (after approximately 5 million iterations).

For purposes of comparison, a Simulated Annealing solution has also been implemented (Richards et al, 2010). At 15000 iterations it yields a width error of 24.7. It also achieves a best width error of 17.24 (but this took 220 million iterations).

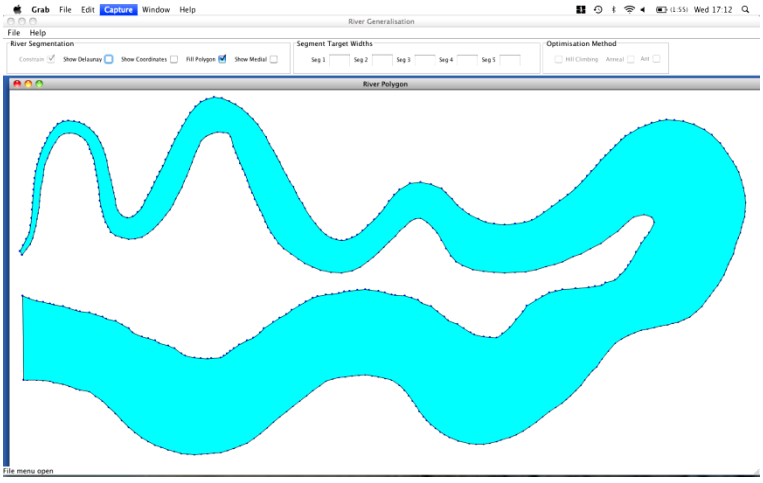



Figure 5 - Simple river polygon.

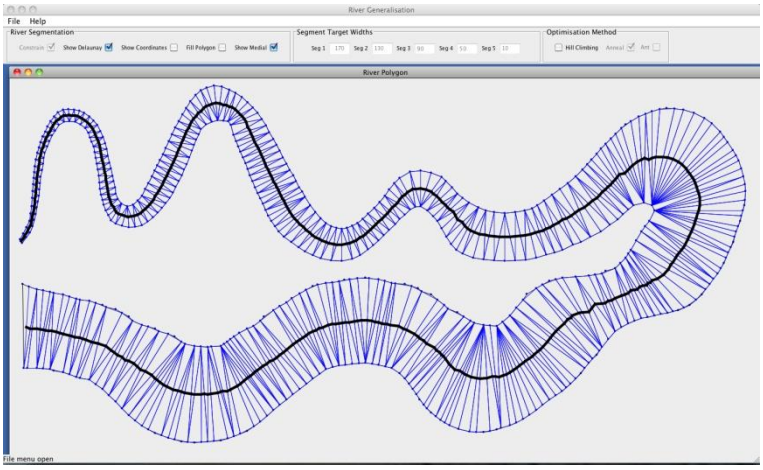



Figure 6 - Triangulated river polygon and centreline.

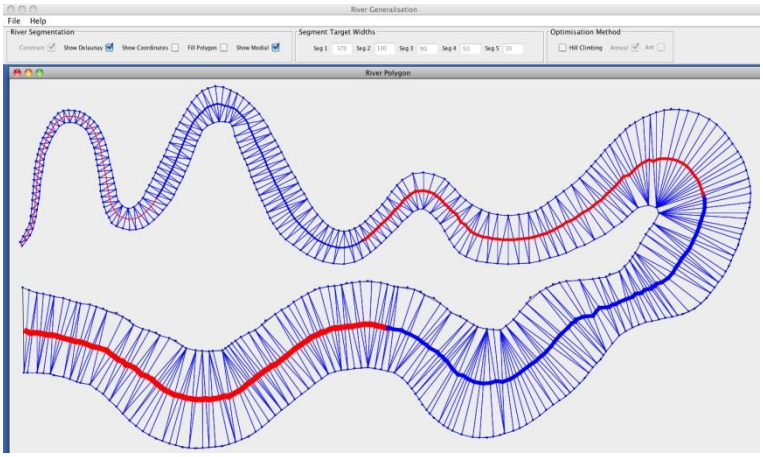



Figure 7 - Original segmentation.

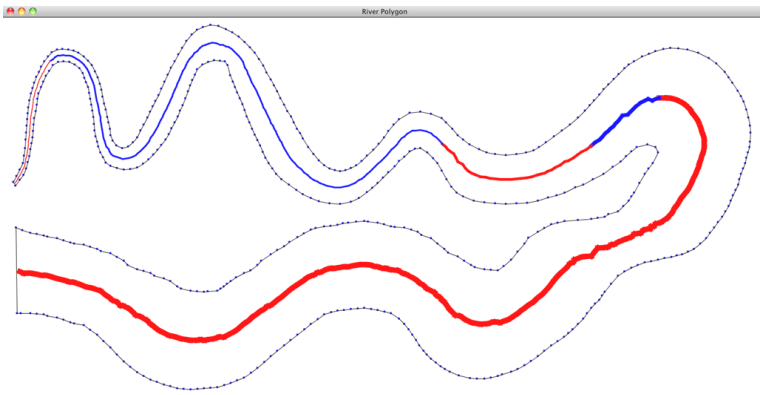



Figure 8 – Segmentation produced by ACO after 15000 iterations.

## 4. Network Schematization

Perhaps the most well known example of a schematic map is the London Tube map designed by Harry Beck (see http://www.tfl.gov.uk/tfl/maps-home.shtml for this and many hundreds of

other examples). The types of schematic maps dealt with in this paper have the following properties:

(i) They are derived from network data sets consisting of polylines, edges and vertices;
(ii) Polylines are simplified to their most elementary shapes;
(iii) They are topologically equivalent to the input network;
(iv) If possible, edges should lie in horizontal, vertical or diagonal direction;
(v) If possible, edges should have length greater than some minimum length (effectively increasing map scale in congested areas).

This paper addresses points (iii)-(v), considering them an optimization problem. Given an input network (pre-simplified using a suitable line generalization algorithm), an alternative state can be obtained by displacing one or more of the network vertices, resulting in re-orientation, shortening and lengthening of edges. The search space being examined is the set of all possible states of the input network. Each state can be evaluated in terms of how closely it resembles a schematic map (i.e. meets a set of constraints based on (iii)-(v)). However, finding the best state by exhaustively generating and evaluating all possible states in not possible, as for any realistic data set the search space will be excessively large. An ACO algorithm for producing schematic maps for network data has therefore been developed.

*4.1 The algorithm*

Each vertex in the network is assigned two matrices– a displacement matrix and a pheromone matrix. The displacement matrix is centred over the original location of the vertex and its cells represent all possible locations into which the vertex can move. Cell size governs the minimum distance a vertex can be displaced. Cell size together with matrix size (the number of cells) determines the maximum distance a vertex can move. Each displacement matrix cell has a corresponding cell in the pheromone matrix. The value of a pheromone cell represents pheromone strength at its corresponding location at any given time; to begin, all pheromone matrix values are initialized to a pre-determined value.

ACO is an iterative process involving a colony of artificial ants working in parallel. Ant colony size is an input parameter to the algorithm (there is no set value). For each iteration each ant starts with the original network (no vertex displacement) and builds its own solution by performing a fixed number (e.g. 1000) of vertex displacements. After each displacement the network is evaluated (against the constraints) and assigned a cost. For each displacement, a vertex is randomly selected and allowed to move from its current matrix location to an adjacent matrix cell. The direction of vertex movement is chosen by a so-called state transition rule, which is influenced by the vertex's associated pheromone matrix (higher values encourage movement) and additional heuristic information (e.g. immediate cost benefit). Within each iteration, a displacement triggers a corresponding reduction of pheromone value (this so-called local update encourages a more complete exploration of the search space). Note that for each vertex all ants are accessing and updating the same pheromone matrix. At the end of each iteration each ant will have produced a solution. The best solution (lowest cost) is used to globally update all pheromone matrices to strengthen pheromone values along the paths from original vertex locations to their new locations in this best solution. This encourages better moves during the iterations that follow. The process repeats until stopping conditions (e.g. maximum number of iterations, maximum time, acceptable cost, etc.) are met.

*4.2 Initial results*

The ACO algorithm has been implemented using Java. Initial experiments have been carried out using OSCAR road centre line data for the St. David's area of West Wales. The original test data consisted of 205 edges, made up from a total of 187 vertices. This data is pre-generalized using the ArcGIS Simply Line tool; with point remove and topological error check options selected this makes use of an enhanced version of the Douglas-Peucker algorithm. A weed tolerance value of 50(m) is used in these experiments, resulting in 67 edges made up from a total of 59 vertices. The simplified data (Figure 9) acts as input to the ACO algorithm. The initial cost (in terms of constraints) for the input network is 640.5.

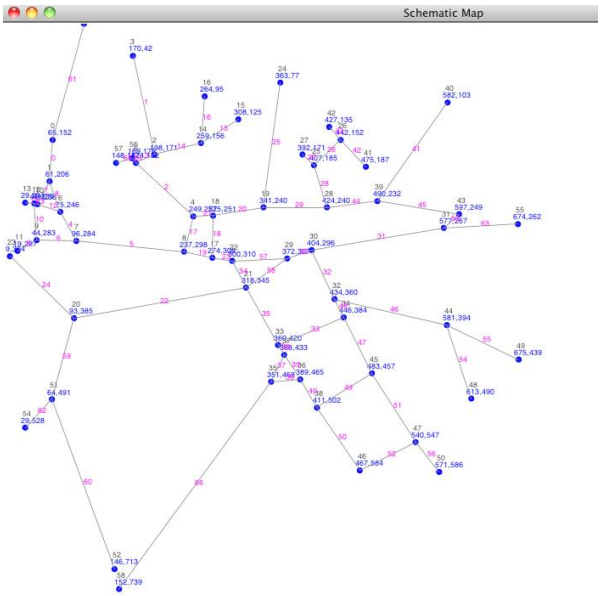



Figure 9 – Input road network. Cost = 640.5.

Figure 10 shows output produced by ACO after a total of 976,000 vertex displacements (in less than 2 seconds). The cost has been reduced to 99.5. For purposes of comparison, a Simulated Annealing (SA) solution (based on Ware et al, 2006) has also been implemented in Java. The best SA solution generated to date (shown in Figure 11) has a cost of 171.1 (after 3,450,000 vertex displacements). It was also noted that after 25,000 vertex displacements, ACO cost had reduced to 179.1 and SA cost had reduced to 333.0.

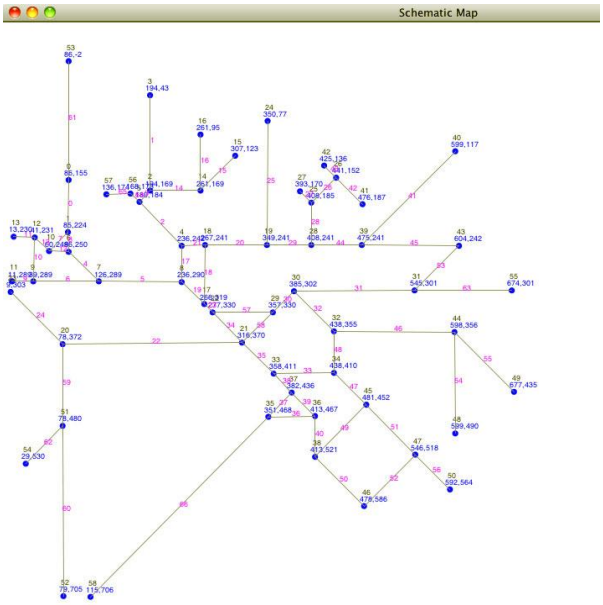



Figure 10 – Schematic map produced by ACO. Cost = 99.5, number of vertex displacements = 976,000.

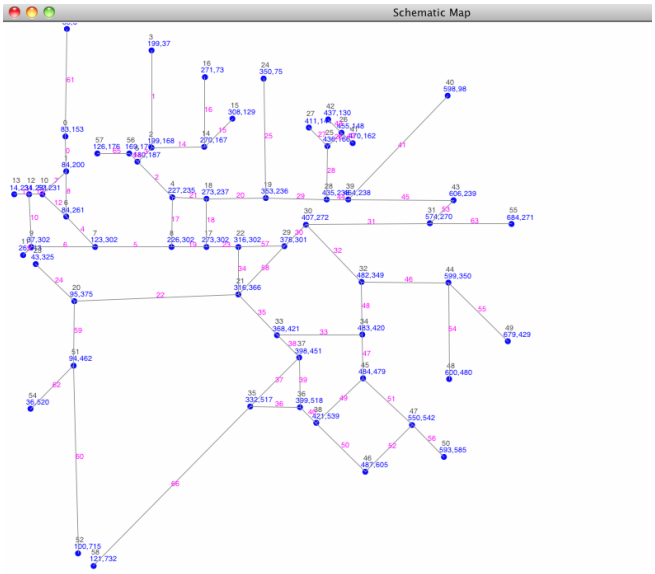



Figure 11 – Schematic map produced by SA. Cost = 171.1, number of vertex displacements = 3,450,000.

## 5. Conclusion

This paper has presented an ACO solution to the particular problems of river segmentation and network schematization. It has shown that ACO can be successfully applied to a segmented river centreline for the purpose of reducing width error. When compared to a Simulated Annealing solution, ACO is able to reduce width error at a faster rate (i.e. in less iterations). It should be noted that both algorithms achieve the same overall best result – which maybe suggests that the global optimum has been found. Future work will concentrate on testing the ACO on more realistic data sets. It is also noted that measuring segmentation quality on width error alone might be too simplistic an approach. The intention is also to apply ACO to other cartographic problems. Some work has been carried out on developing

an ACO solution to the well known Schematic Map problem (Ware et al, 2006) and the initial results, reported here, indicate that ACO again outperforms Simulated Annealing (both in terms of quality of result and processing times).

## 6. References


Dorigo, M., Maniezzo, V. and Colorni, A., 1996, Ant System: Optimization by a Colony of Cooperating Agents, IEEE Transactions on Systems, Man, and Cybernetics–Part B, Volume 26, Number 1, p. 29–41.

Goss, S., Aron, S., Deneubourg, J.-L. and Pasteels, J.M.,1989, Self-organized shortcuts in the Argentine ant. Naturwissenschaften, Volume 76, p. 579–581.

Jones, C.B., Bundy, G.Ll. and Ware, J.M., 1995, Map generalisation with a triangulated data structure, Cartography and GIS, Volume 22, Number 4, p. 317-331.

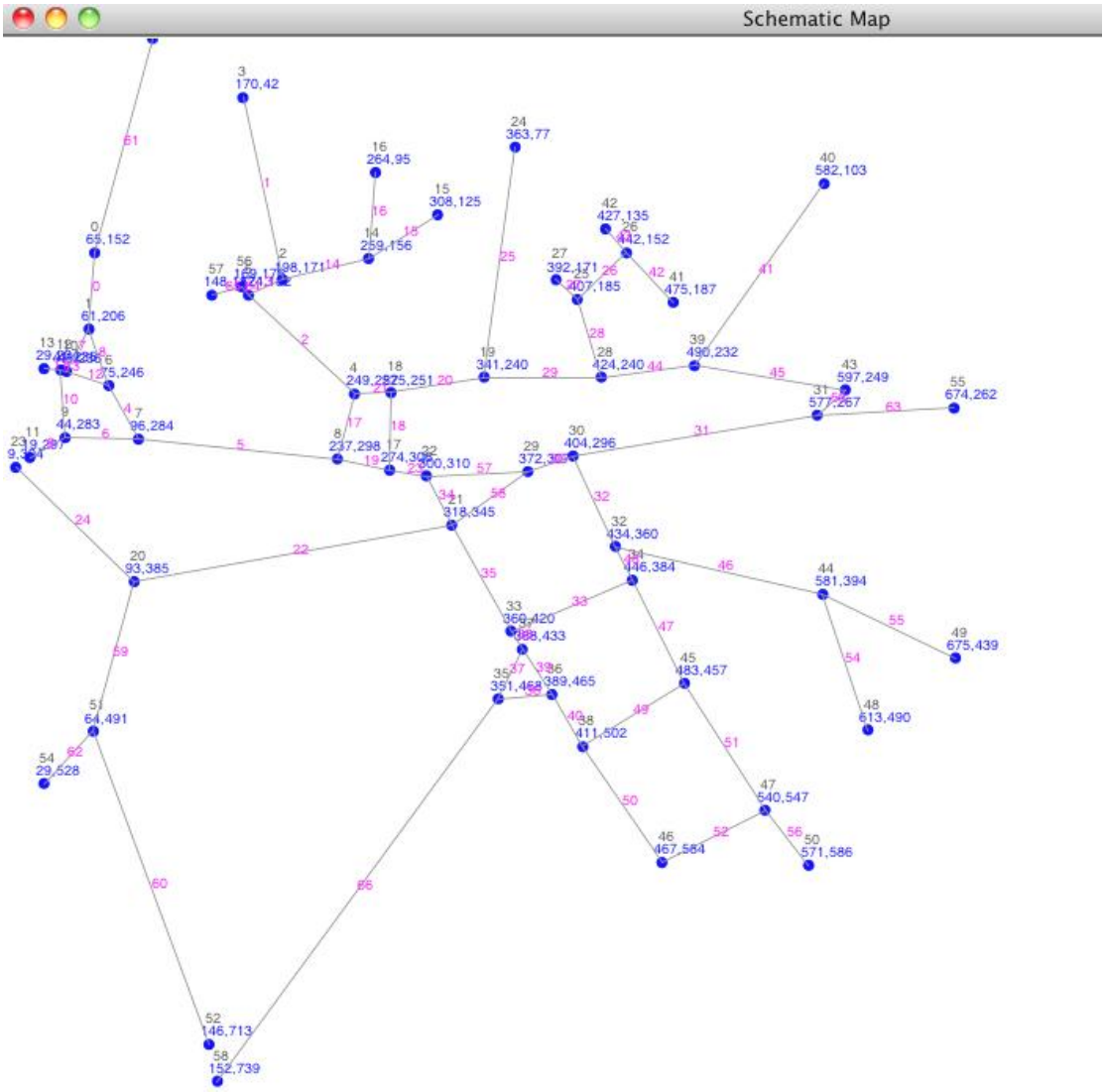
Regnauld, N. and Mackaness, W., 2006, Creating a hydrographic network from its cartographic representation: A case study using Ordnance Survey MasterMap data, International Journal of GIS, Volume 20, Number 6, p.611–631.

Richards, N., Ware, J.M., Thomas, N. and Ware, J.A., 2010, Automated map generalization: application of simulated annealing to river symbolization, accepted for 2010 International Conference on Artificial Intelligence (Las Vegas, July 2010).

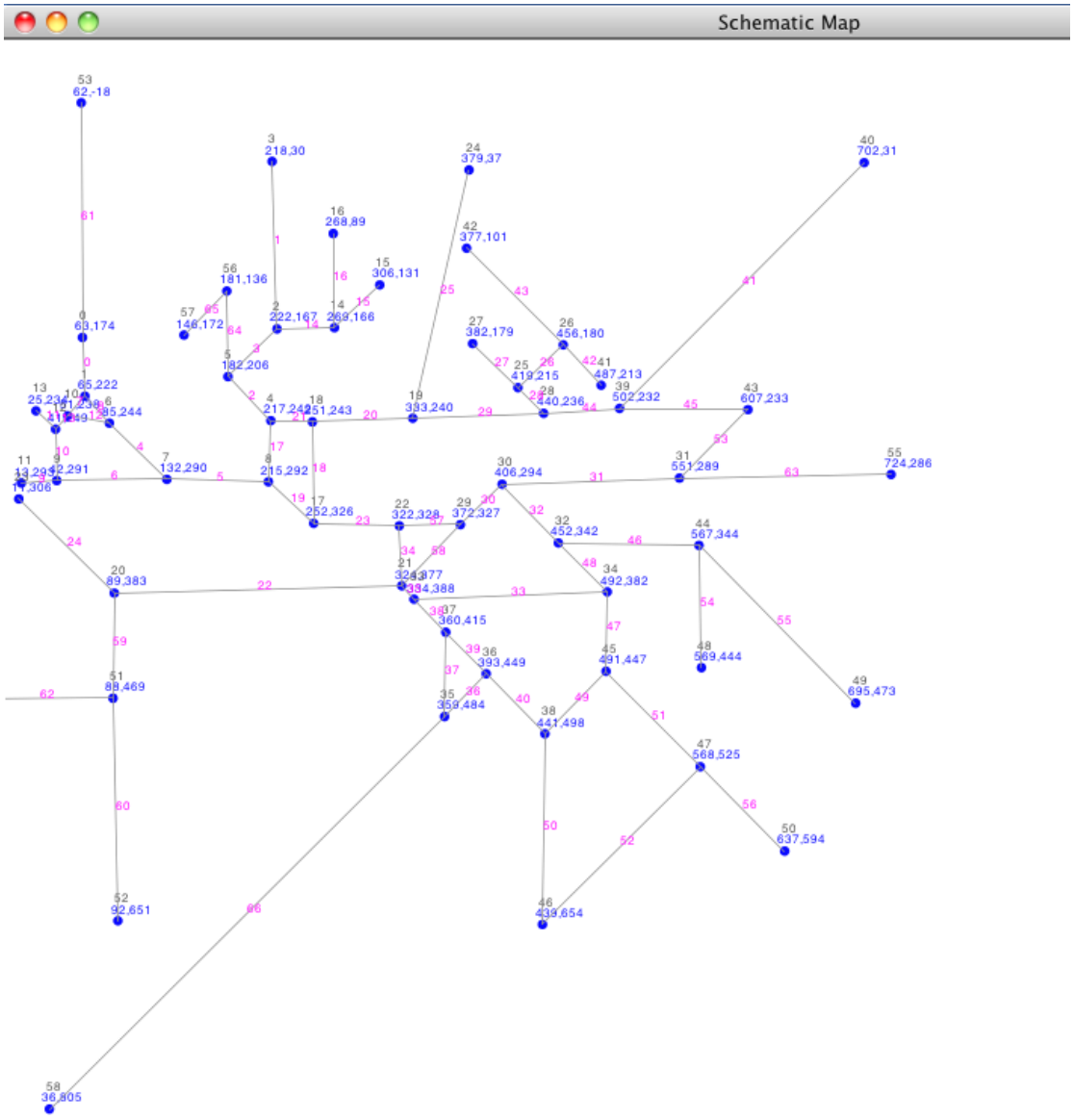
Ware, J.M., Anand, S., Taylor, G.E. and Thomas, N., 2006, Automatic Generation of Schematic Maps for Mobile GIS Applications, Transactions in GIS, Volume 10, Issue 1, p. 25-42.

**Appendix 1 – ACO and Simulated Annealing Applied to Schematic Map Problem**

Original Road Network – cost (in terms of schematic map constraints – see Ware et al, 2006) = 640.



Schematized Road Network Produced Using ACO – cost = 99, number of iterations = 900000.



Schematized Road Network Produced Using Simulated Annealing – cost = 171, number of iterations = 3500000.