Solving a unique Shortest Path problem using Ant Colony Optimisation

Daniel Angus

Abstract. Ant Colony Optimisation (ACO) has in the past proved suitable to solve many optimisation problems. This research explores the ability of the ACO algorithm to balance two quality metrics (length and cost) in its decision making process. Results are given for a preliminary investigation based on a series of shortest path problems. It is concluded that, for these problems at least, the solution quality and time to solution make ACO competitive as an optimisation technique for shortest path problems in which multiple factors such as cost and length are involved.

1 Introduction

Optimisation of common processes presents an ongoing challenge to researchers and practitioners alike. Often involving a large number of possible solutions these problems usually arise in large industries such as telecommunications, transportation and electronics where even slight increases in solution quality can translate to increased company profit, lower consumer prices and improved services. As a result of this, numerous optimisation techniques have been studied, developed and refined.

The traditional operations research techniques of branch and bound, cutting planes and dynamic programming have been widely used, but can be computationally expensive for large and complex problems. As a result, newer metaheuristic search algorithms such as simulated annealing [10], tabu search [6] and genetic algorithms [7] have been applied as they generally find good solutions in a moderate amount of computational time.

The particular optimisation problem undertaken in this research is a shortest path problem that has been characterised by the seminal work of Dijkstra [11]. Using a form of branching Dijkstra was able to exhaustively search and hence solve the shortest path problem.

This research aims to use the ant colony optimisation algorithm [1 - 5][9], to solve a unique shortest path problem. The aim is not to exhaustively search all possible solutions but to find good solutions in a small amount of computation time.

2 Problem Definitions

2.1 Shortest Path Problem

A shortest-path problem involves a weighted, possibly directed graph described by a set of edges and vertices. Given a start vertex, the goal is to find the shortest existing path between the start vertex and any of the other vertices in the graph.

Each path, therefore, will have the minimum possible sum of its component edges' weights.

Formally, what one tends to think of as the ‘length’ of an edge is known as its ‘weight’. Thus, a graph whose edges are all of equal length is *unweighted*, whereas a graph with edges of differing lengths is *weighted.* The term ‘weight’ is used because graphs are not limited to representing locations on a plane or in space; consequently edges could represent time, cost, and so on, generally a quantity which is to be kept minimal when going from any vertex to another.

There exist many techniques for solving the shortest path problem, some of the better known algorithms are Dijkstra’s [11] and Bellman-Ford[12].

2.2 Travelling Salesman Problem

While not actually studied in this investigation the Traveling Salesman Problem (TSP) will be used to describe the Ant Systems meta-heuristic and so needs to be included. The TSP is also a possible project extension as described in Section 6.4.

The TSP is a popular path optimisation problem described as:

Given a set of n vertices and weights for each pair of vertices, find a roundtrip of minimal total weight visiting each vertex exactly once.

The set of n vertices can be represented as geographical coordinates, Cartesian coordinates or even as a matrix of weights between nodes. How the vertices are represented is not important as long as a weight between every node can be calculated.

Most popular TSP data sets contain a 2-Dimensional coordinate set. Since we are only concerned about the weight between nodes and not the nodes themselves there is no real difference between a 2D and a 3D representation, only that one more variable is included in the 3D distance calculation.

A large catalogue of TSP problems and literature can be found at TSPLIB [8].

3 Class of Test Problem Used

The test problems constructed for this analysis are discrete artificial representations of geographical landscapes including valleys, hills and plateaus. All test problems are completely artificial and have been created to exploit perceived strengths and weaknesses in the algorithm.

Each test problem created for this analysis has been purposefully designed to gauge how efficient the algorithm is on a range of non-challenging to highly challenging terrains. Note that in some situations it may be desirable to traverse a large geographical distance to minimise the energy expended[1], and that the shortest path (distance-wise) is not always the most energy efficient path, as the short path may include many hills or difficult terrain.

[1] This factor is derived from the author's passion of backcountry skiing which was the original inspiration for the project.

3.1 Nature of Terrain

Each terrain data set consists of a set of vertices and edges. The vertices are represented by three Cartesian coordinates (x,y,z) with the x and y coordinates being evenly distributed at unit intervals apart. While the x and y coordinates are distributed in an even grid, the z coordinate (altitude) can assume any value, positive or negative. The edges connect each vertex to neighbouring vertices as shown in Figure 1. The terrain is asymmetric, meaning that edge (i,j) ≠ edge (j,i).

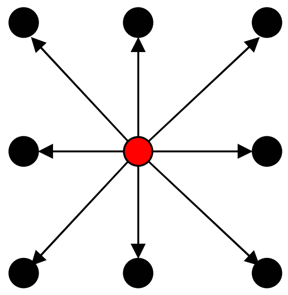


Figure 1: Edge connectivity

Boundary conditions are simple with boundary vertices only being connected to interior vertices as shown in Figure 2.

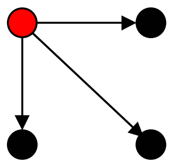


Figure 2: Boundary conditions

3.2 Scoring Results

With positive z as a reference, the elevation angle (θ) that each edge subtends can be calculated. This angle can be used as a basis for the cost function, which is described in Equation 1 and Figure 3. The cost function described performs a simple operation on the elevation angle and returns a number representing the cost per unit length (energy expended/recovered) within the range [-0.2,1].

$$f(\theta) = 1 - \frac{0.6}{90}\theta \text{ , } 0 \le \theta \le 180$$

Equation 1: Cost function

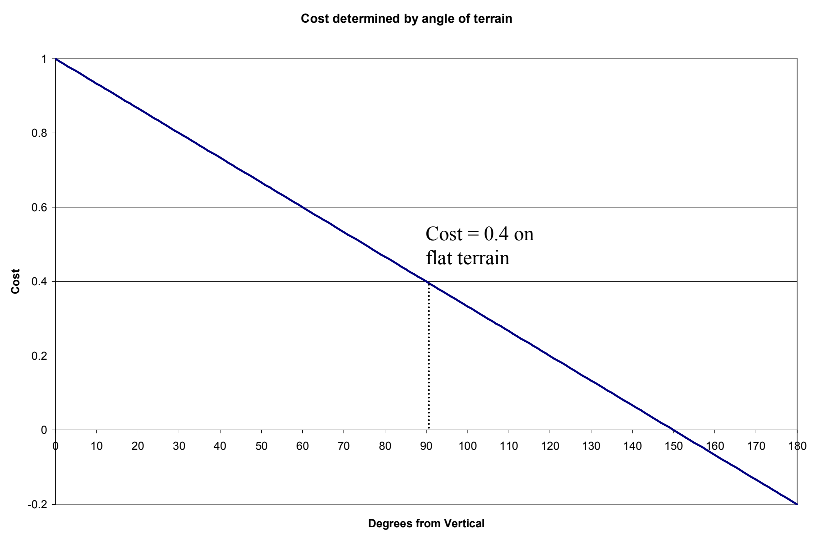


Figure 3: Cost function

Note that for horizontal terrain (90 degrees from vertical) the cost is still positive (energy is expended), and that it isn't until the angle reaches 60 degrees past horizontal that the cost is actually negative, meaning that energy is being recovered.

$$\textit{edge weight} = \textit{edge cost} \times \textit{edge length}$$

Equation 2: Edge weight calculation

$$\textit{total path energy} = \sum \textit{edge weight}_{\textit{edges included in tour}}$$

Equation 3: Total path energy calculation

The total path energy (Equation 3), which is the measure of success, is calculated by summing the weights of all edges included in a tour (Equation 2). In all problems it is desired to choose a path that minimises the total path energy.

3.3 Exhaustively Searching

It was necessary to exhaustively search each terrain set to determine a benchmark upon which the performance of the ant colony algorithm could be measured. The exhaustive search technique developed was a depth-first search algorithm.

The depth-first search algorithm is a class of branching algorithm that evaluates each branch in turn, rather than evaluating all the branches simultaneously, as the breadth-first search does. By evaluating each branch in turn the depth-first search algorithm is less memory intensive and therefore does not fail on larger data sets. The results of the exhaustive search are included in Section 6: Results and Discussion, where available.

4 Ant Colony Optimisation

ACO is modelled on the foraging behaviour of Argentine ants. The seminal work by Dorigo [1] showed that this behaviour could be used to solve discrete optimisation problems. This section gives a brief overview of the ant colony mechanics using the Ant Systems (AS) meta-heuristic and the travelling salesperson problem (TSP) together with other applications.

4.1 Ant Systems (AS) as applied to the TSP

In AS all ants construct candidate solutions based on two heuristics: pheromone and a problem dependent heuristic.

For the TSP, the problem dependent heuristic is visibility (η) and is defined as the inverse of the distance (d) between vertices as shown in Equation 4.

$$\eta = 1/d$$

Equation 4: Visibility

To begin with all edges are initialised with an initial amount of pheromone (τ_0).

After initialisation each ant constructs a path by choosing vertices based on pheromone levels (τ) and visibility (η) until they have visited all cities exactly once.

The choice fof the next vertex is a probabilistic one. An edge with a large weight increases the probability of that edge being chosen; therefore closer vertices have an increased probability of being chosen, as do vertices connected by edges with higher pheromone levels.

Each ant maintains a tabu list ensuring that cities that have already been visited have a zero chance of being visited again.

Two parameters, α and β, control the importance of edge pheromone intensity and visibility respectively in the probabilistic selection of the next vertex to visit. If $\alpha = 0$ the algorithm behaves as a standard greedy algorithm, with no influence from pheromone. If $\beta = 0$ only pheromone amplification will occur and the distance between cities has no influence on the choice. Usually a trade-off between these two factors is best.

When all ants have completed one tour, the amount of pheromone is updated. On every edge the current pheromone will be decreased by a set percentage known as the decay constant (ρ). Each ant deposits pheromone across edges contained in its previous tour with the amount of pheromone deposited controlled by the total length of the tour, i.e. the shorter the tour, the more pheromone deposited. The amount of new pheromone introduced is influenced by the constant parameter (Q).

This process continues until a certain condition is satisfied (number of iterations, amount of CPU time, certain solution found). The number of ants (m) should be equal to the number of cities (n) in this algorithm.[2] Too many ants would quickly

[2] According to Dorigo, M., V. Maniezzo, and A. Colorni. "The Ant System: Optimisation by a Colony of Cooperating Agents". *IEEE Trans. Syst. Man Cybern*. 1996

reinforce sub-optimal solutions, and too few would not efficiently produce results based on pheromone decaying too quickly.

The basic AS algorithm is:

```
/* Initialization */
For every edge $(i,j)$ do
  $\tau_{ij}(0) = \tau_0$
End For
For $k = 1$ to $m$ do
  Place ant $k$ on a randomly chosen city
End For
Let $T^+$ be the shortest tour found from beginning and $L^+$ its length
/* Main loop */
For $t = 1$ to $t_{\max}$ do
  For $k = 1$ to $m$ do
    Build tour $T^k(t)$ by applying $n - 1$ times the following step:
    Choose the next city $j$ with probability

        $p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in J_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta}$ ,

    where $i$ is the current city
  End For
  For $k = 1$ to $m$ do
    Compute the length $L^k(t)$ of the tour $T^k(t)$ produced by ant $k$
  End For
  If an improved tour is found then
    update $T^+$ and $L^+$
  End If
  For every edge $(i,j)$ do
    Update pheromone trails by applying the rule:
    $\tau_{ij}(t) \leftarrow (1 - \rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t)$ where

        $\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^k(t)$ ,

        $\Delta\tau_{ij}^k(t) = \begin{cases} Q/L^k(t) & \text{if } (i,j) \in T^k(t); \\ 0 & \text{otherwise}, \end{cases}$
  End For
  For every edge $(i,j)$ do
    $\tau_{ij}(t + 1) = \tau_{ij}(t)$
  End For
End For
Print the shortest tour $T^+$ and its length $L^+$
Stop
```

5 Shortest Path Ant Colony Optimisation

The Ant Systems (AS) meta-heuristic in Section 4.1 requires significant modification to be applied to the shortest path problem described in Section 3, one reason being that decisions now have to be based on three parameters, cost, visibility and pheromone. Based upon the AS meta-heuristic a new algorithm, the Shortest Path Ant Colony Optimisation (SPACO) algorithm was developed.

Each of the metrics used in the SPACO algorithm are described in detail in this section, along with the equations for combining all three metrics into a single probability figure.

5.1 Cost selection metric

The cost function described in Section 3.2 is bounded between [-0.2,1], which is inappropriate for direct usage as a selection metric due to the negativity, which could give rise to negative selection probabilities. Therefore a simple transformation was applied to the cost function as shown in Equation 5.

$$\boldsymbol{desirability} = \frac{1}{2^{cost}}$$

Equation 5: Desirability contribution from cost function

This equation transforms the cost function into a desirability scale of range [0.5,1.15]. This means that edges with a higher cost will have a lower desirability of being selected, and conversely edges with low cost will have higher desirability.

The contribution of the cost function to the calculation of the total desirability depending on the angle of the terrain is shown in Figure 4.

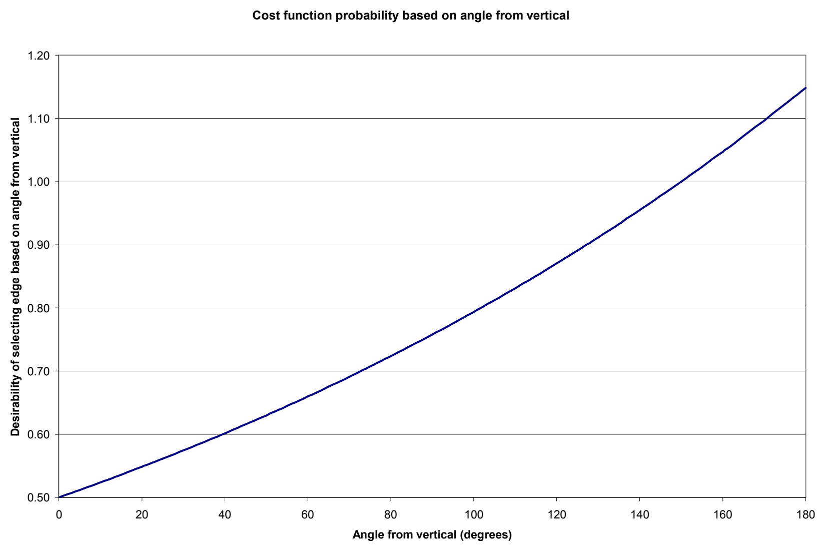


Figure 4: Cost function desirability contribution

5.2 Visibility

The visibility is the metric that is intended to influence each ant in moving towards the target vertex based on a minimal straight-line distance. This differs from the TSP in which visibility only concerns the next vertex.

Visibility is problem specific, for this problem visibility is based upon the distance from the current position and the length to the target as shown in Figure 5.

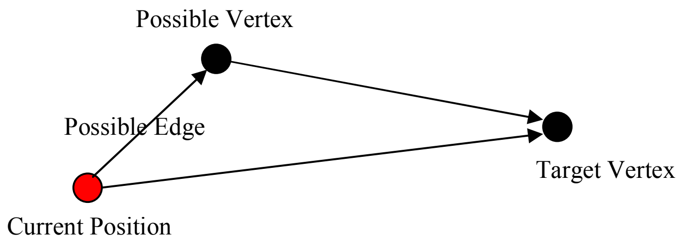


Figure 5: Visibility Definition

The desirability of traversing a possible edge based on the visibility metric is calculated according to Equation 6.

$$\textbf{\textit{visibility}} = \left(\frac{\textbf{\textit{distance from current position to target vertex}}}{\textbf{\textit{distance from possible vertex to target vertex}}} \right)$$

Equation 6: Visibility Calculation

The distance between the current position and the target vertex determines the range of values that the visibility can take. Figure 6 illustrates three simple cases where the target vertex is located to the right at a distance as shown in the centre of each table (50, 10 and 2 respectively). The distances calculated (with the current position in bold) translate into the corresponding visibility values directly below.

Distance from Target Vertex		
51.010	50.010	49.010
51.000	**50.000**	49.000
51.010	50.010	49.010
Visibility based on distances above		
0.980	1.000	1.020
0.980	**Current Position**	1.020
0.980	1.000	1.020

Distance from Target Vertex		
11.045	10.050	9.055
11.000	**10.000**	9.000
11.045	10.050	9.055
Visibility based on distances above		
0.905	0.995	1.104
0.909	**Current Position**	1.111
0.905	0.995	1.104

Distance from Target Vertex		
3.162	2.236	1.414
3.000	**2.000**	1.000
3.162	2.236	1.414
Visibility based on distances above		
0.632	0.894	1.414
0.667	**Current Position**	2.000
0.632	0.894	1.414

Figure 6: Example visibility calculations

From the examples shown in Figure 6 it is clear that the visibility heuristic will have a stronger influence depending on the distance from the target vertex. This characteristic is not undesirable, as it may be intuitive for an ant to move quickly towards the target vertex when it appears in its immediate vicinity.

On larger problems (in terms of the distance from start to end vertices) the visibility metric may need to be scaled so as it can be of significant influence to the final calculated edge desirability. This effect can be seen in Figure 6 where the raw visibility for a distance of 50 ranges between [0.98, 1.02].

5.3 Pheromone

The pheromone metric is the defining property of Ant Colony Optimisation Techniques, and such, it has often been the topic of previous research. In attempting to solve the shortest path problem it was decided to approach the pheromone metric from first principles.

It was also decided that the decay and update rules were to be constructed in such a way as to hold the total system pheromone constant. This was introduced since it had been observed in prior experimentation that the total system pheromone amount would often decay or explode as time evolved. If it were possible to hold the total system pheromone at some value then the only concern would be the distribution of pheromone within the system not the actual total amount.

An iteration is defined as each ant in the population traversing an edge based on the probabilistic selection algorithm.

5.3.1 Initial Pheromone

The initial pheromone density and the total number of edges determines the total amount of pheromone contained within the system for the life of the problem. A unit of pheromone is placed on each edge at initialisation, and therefore the total pheromone (in units) contained within the system is equal to the number of edges.

5.3.2 Pheromone Decay

After each iteration, every edge's pheromone is decayed by a set percentage, so that edges with higher pheromone concentrations lose more pheromone than edges with lower concentrations (Equation 7).

$$\boldsymbol{pheromone_{ij} = (1 - decay\ constant) \times pheromone_{ij}}$$

Equation 7: Pheromone decay rule

5.3.3 Pheromone Update

Many techniques allow ants to complete tours before laying down pheromone, so that the quality of tour can be used to determine the pheromone strength to be updated along the tour path.

The pheromone update rule introduced in this algorithm is akin to biology with an ant laying down of pheromone at a preset rate as it navigates a problem. After each iteration, an ant adds the preset amount of pheromone to the edge it has just traversed.

$$\textbf{\textit{If edge has been traversed then : pheromone}}_{ij} = \textbf{\textit{pheromone}}_{ij} + \textbf{\textit{update}}$$

Equation 8: Pheromone update rule

If two or more ants, in the same iteration, traverse an edge, it will receive an update directly proportional to the number of ants that traversed it in that iteration.

5.3.4 Pheromone Decay versus Pheromone Update

In order to achieve total system pheromone stability Equation 9 must be satisfied.

$$\textbf{\textit{total system pheromone}} = \frac{\textbf{\textit{number of ants}} \times \textbf{\textit{update constant}}}{\textbf{\textit{decay constant}}}$$

Equation 9: Total system pheromone stability condition

5.4 Edge Selection Algorithm

When deciding which edge to traverse next an individual ant relies on a probabilistic selection algorithm. This algorithm combines all of the metrics of an edge into a single quality measure. This calculated value is the desirability of moving along this edge at the next time interval. This figure is compared to the other available edges in order to gauge its relative strength against another edge.

When combining metrics the relative importance of an individual metric can be asserted or diminished relative to the other metrics. This process is usually accomplished through the use of scaling factors in the form of powers or constants boosting or shrinking an individual metric's raw value.

The combination of metrics is the cornerstone to the success or failure of this technique and the combined metric must provide an accurate reflection of the perceived strengths or weaknesses of an edge.

5.4.1 Product Combination

To date the majority of Ant Colony Optimisation techniques combine their metrics in a product fashion. In this study the metrics were also combined in product form as shown in Equation 10.

$$\text{Prob}_{ij} = \frac{\text{pheromone}_{ij}^{\alpha} \bullet \left(\frac{1}{2^{\text{cost factor}_{ij}}} \right)^{\beta} \bullet (\text{visibility})^{\chi}}{\sum \left[\text{pheromone}_{ij}^{\alpha} \bullet \left(\frac{1}{2^{\text{cost factor}_{ij}}} \right)^{\beta} \bullet (\text{visibility})^{\chi} \right]}$$

Equation 10: Product rule

The product rule has been proven to work well when combining two metrics, however as it was unknown how well it would perform with three metrics therefore an alternative metric combination rule was developed in this project: the vector addition rule.

5.4.2 Vector Addition Combination

The vector addition rule is exactly that, each heuristic is treated as a component of a vector and the magnitude of this vector represents the value of the metric (Equation 11).

$$\text{Prob}_{ij} = \frac{\sqrt{\left(\text{pheromone}_{ij}^{\alpha}\right)^2 + \left(\left(\frac{1}{2^{\text{cost factor}_{ij}}}\right)^{\beta}\right)^2 + \left(\left(\text{visibility}\right)^{\chi}\right)^2}}{\sum\left[\sqrt{\left(\text{pheromone}_{ij}^{\alpha}\right)^2 + \left(\left(\frac{1}{2^{\text{cost factor}_{ij}}}\right)^{\beta}\right)^2 + \left(\left(\text{visibility}\right)^{\chi}\right)^2}\right]}$$

Equation 11: Vector rule

5.4.3 Product versus Vector Combination

To illustrate the differences between the combination techniques consider a simple dual metric system with visibility and pheromone metrics. If the pheromone is bounded between [0,2] and visibility between [0,1], Figure 8 and Figure 7 are the desirability surfaces obtained using the vector and product rules.

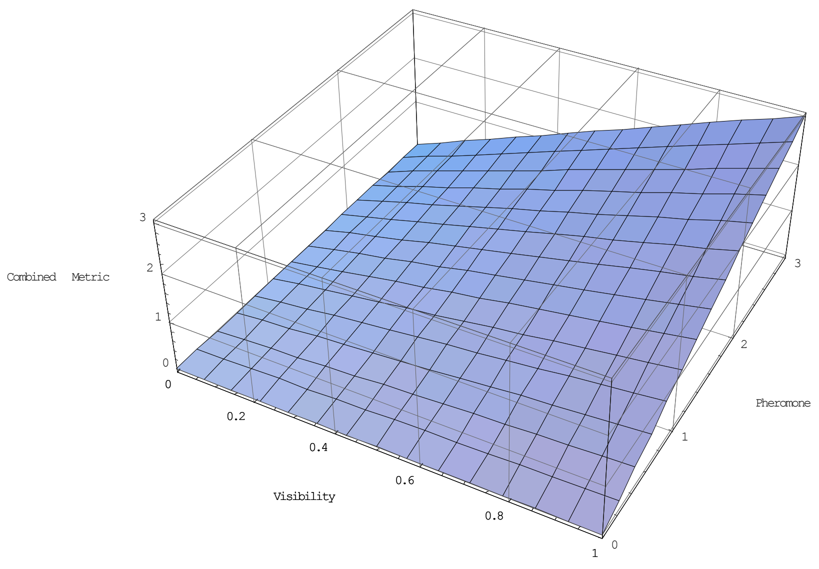


Figure 7: Product rule desirability surface

Figure 8 shows how even when the pheromone metric is zero the visibility metric is still able to contribute towards the final desirability value. This is not true in Figure 7, where the entire desirability approaches zero when either metric approaches zero.

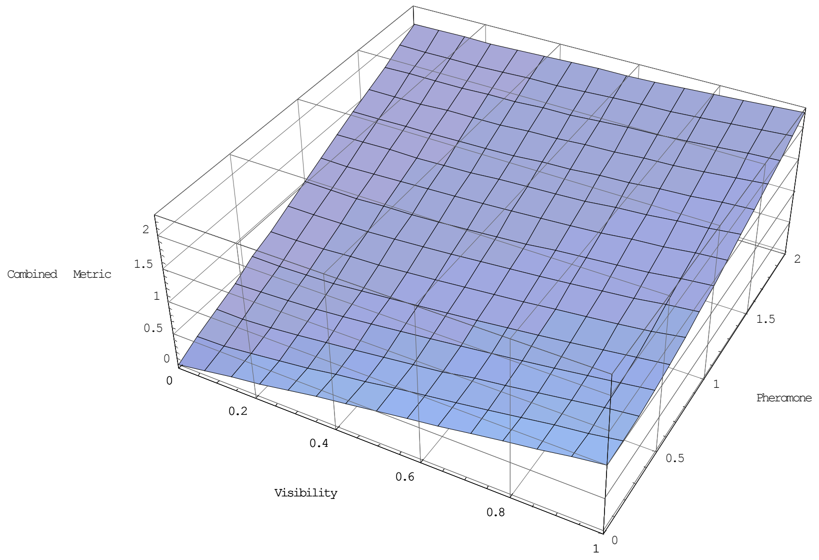


Figure 8: Vector rule desirability surface

It will depend on the application of the algorithm as to which rule would be most appropriate. The multiplication rule is a somewhat greedy selection rule while the vector rule tends to allow for some non-greedy selection characteristics. The anticipated effect that this will have on the final solution quality is that the multiplication rule may give rise to a higher quality of solution, whereas the vector rule will ensure broader search characteristics.

While both techniques have been used in the work reported here, analysis of the results shows that the practice of initialising all pheromone values to a non-zero value, while suitable for the multiplicative technique, handicaps the vector combination approach. This will be discussed further in Section 8.

6 Results and Discussion

6.1 Testing on non-difficult problems - Flat & Mound terrains

It was necessary to test SPACO on smaller problems to ensure that it worked as expected. The flat and mound terrains represent trivially small problems, however the aim was to validate each metric used in the SPACO algorithm.

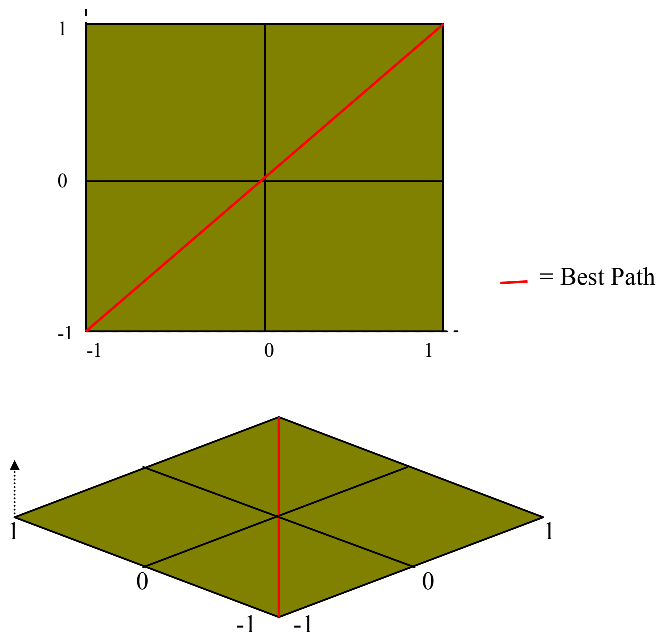


Figure 9: Flat terrain

Flat terrain statistics	
Vertices	9 (3x3 grid)
Edges	40
Minimum Energy Path	1.131
Possible Paths	235

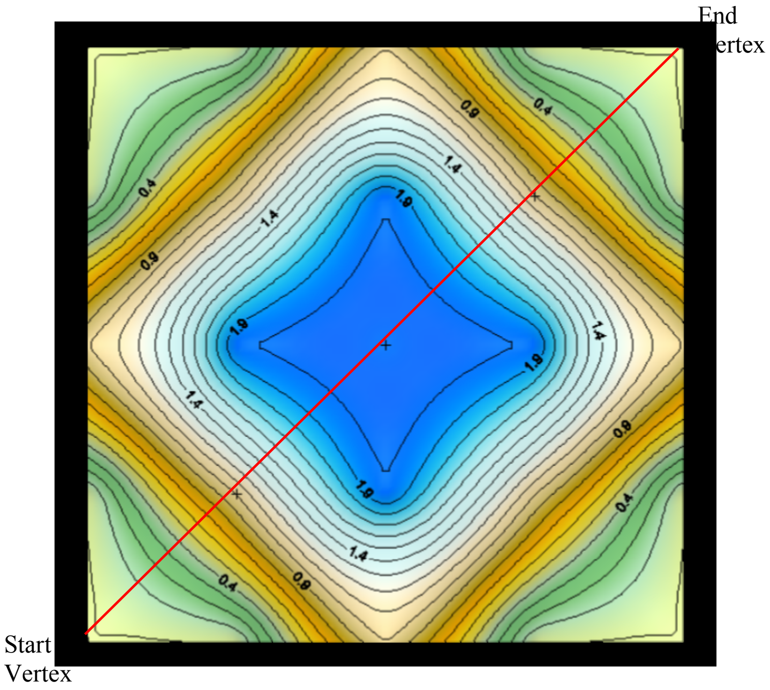
Table 1: Flat terrain statistics

Path Ranking	Path Energy
1	1.131
2	1.366
3	1.366
4	1.366
5	1.366
6	1.366
7	1.366
8	1.600
9	1.600
10	1.600
11	1.600
12	1.600
13	1.600
14	1.931
15	1.931
16	1.931
17	1.931
18	1.931
19	1.931
20	1.931

Table 2: Best 20 paths

This relatively small terrain still contains 235 different path possibilities. Table 2 shows the top 20 paths found using the exhaustive search technique described in Section 3.3.

Using both vector and multiplicative combination techniques the SPACO algorithm successfully found the minimum energy path 100% of the time in 1000 trials. In all trials the SPACO algorithm found the solution in 2 iterations using both multiplicative and vector combination.



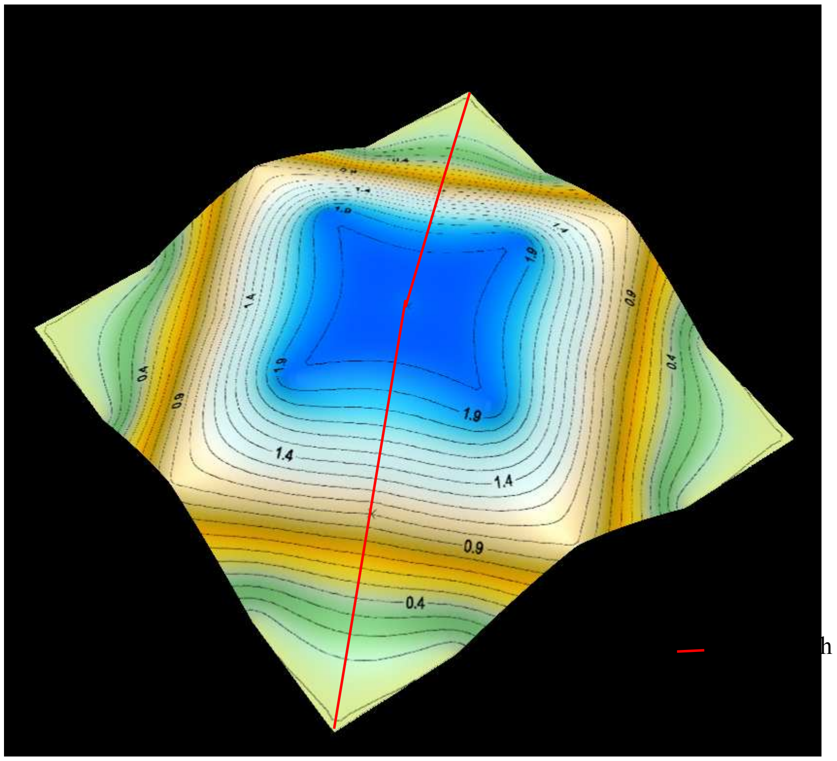


Figure 10: Mound terrain

Mound terrain statistics	
Vertices	25 (5x5 grid)
Edges	144
Minimum Energy Path	2.771
Possible Paths	447,544,629

Table 3: Mound terrain statistics

Using the exhaustive search algorithm the top 20 paths for the mound terrain are shown in Table 4. The time taken to produce these results on a Pentium 3 900MHz PC with 256Mb RAM was 52 hours.

Path Ranking	Path Energy
1	2.771
2	3.027
3	3.027
4	3.027
5	3.027
6	3.061
7	3.061
8	3.061
9	3.061
10	3.063
11	3.063
12	3.083
13	3.083
14	3.086
15	3.086
16	3.142
17	3.142
18	3.245
19	3.245
20	3.266

Table 4: Best 20 paths

Using both vector and multiplicative combination techniques the SPACO algorithm successfully found the minimum energy path 100% of the time in 1000 trials. The SPACO algorithm achieved this result in a mean time of 7 iterations using multiplicative combination and 11 iterations using vector combination.[3]

The results obtained for the flat and mound terrain data sets illustrate that the SPACO algorithm is working as expected and that it can now be applied to more challenging problems.

[3] See Appendices for a full description of parameters used.

6.2 Testing on deceptive problem - Volcano terrain

This particular terrain requires the algorithm to at times move contrary to the direction the visibility heuristic would suggest, in order to find the optimum solution.

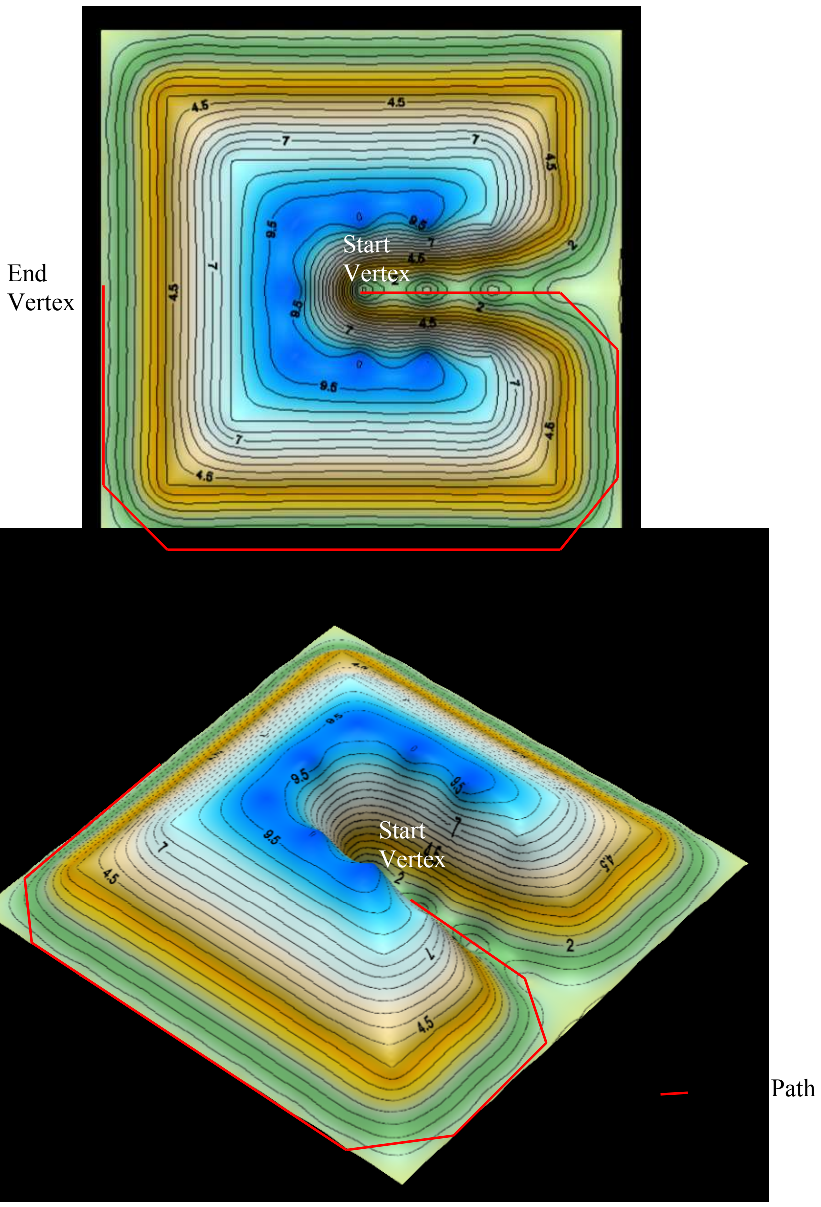


Figure 11: Volcano Terrain

Volcano terrain statistics	
Vertices	81 (9x9 grid)
Edges	544
Minimum Energy Path	7.297
Possible Paths	Approximately = 10^{19}

Table 5: Volcano terrain statistics

The volcano terrain was contrived to determine whether the SPACO algorithm could effectively ignore the visibility heuristic in order to find the optimal solution.

After three weeks of computation time the exhaustive search was still running, due to the large number of possible paths, so therefore the best 20 paths are unable to be produced in this report. However the optimum path of 7.297 was calculated manually for quantitative analysis.

The SPACO algorithm successfully found the minimum energy path 100% of the time in 1000 trials. This result was obtained using a balanced visibility and cost metric importance and using multiplicative metric combination.

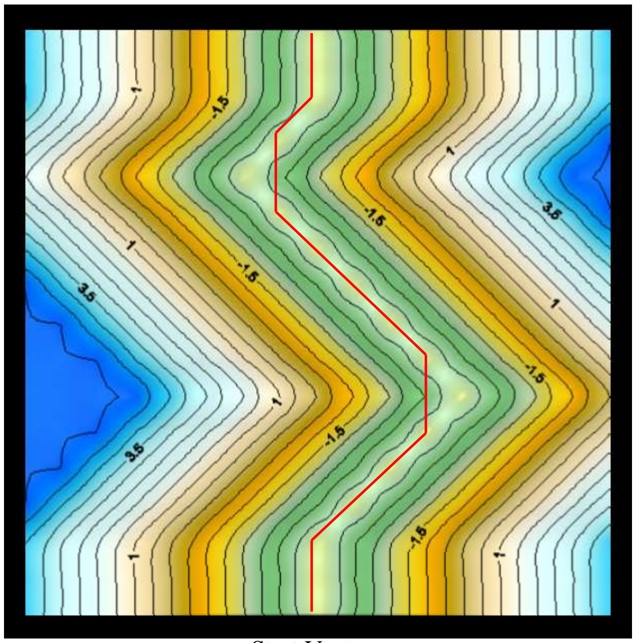
The Vector combination method had little success in solving this terrain, and in 99% of cases converged on a sub-optimal solution that went directly towards the end vertex over the hill rather than moving out and around the hill, which would have led to the optimum solution.

In the multiplicative combination case, even though the algorithm places an equal importance on visibility, which pushes it towards a sub-optimal solution, it can effectively ignore this metric in order to achieve an optimum solution.

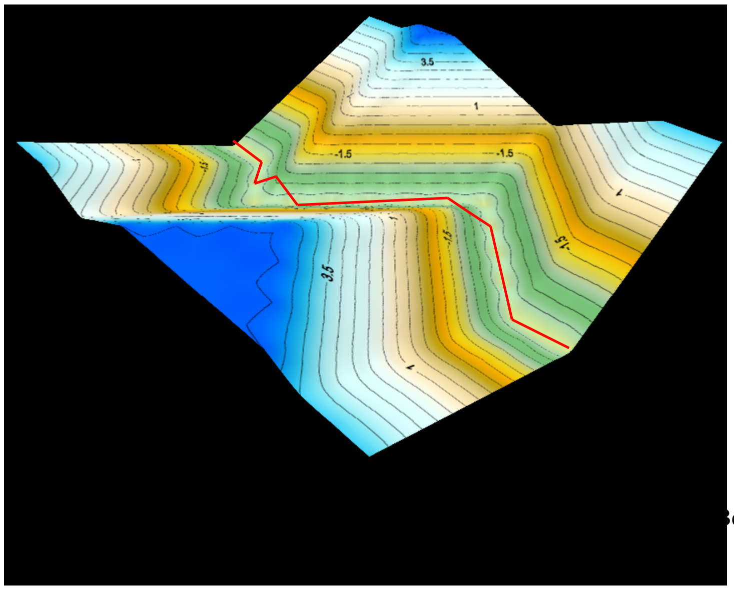
These results highlight the ability of the algorithm to make compromises in which metrics it uses in its decision making while still allowing the metrics to guide it towards a solution.

6.3 Testing on difficult problem - Valley terrain

This terrain was created to determine how well the algorithm is able to trade-off between cost and visibility. While the straightest path is a good path, and staying along the valley floor is also desired, it is the combination of both of these, which creates the optimal result.



Start Vertex



est Path

Figure 12: Valley Terrain

Valley terrain statistics	
Vertices	289 (17x17 grid)
Edges	2112
Minimum Energy Path	8.388
Possible Paths	Approximately = 10^{41}

Table 6: Valley terrain statistics

As with the volcano terrain data set the best 20 paths are unable to be shown in this report due to the large computation time required to produce them. However the optimum path of 8.388 was calculated manually for quantitative analysis.

Problem Parameters				Path Energy				Iterations			
Alpha (Phe)	Beta (Vis)	Gamma (Cost)	Combination Method	Mean	Standard Deviation	Minimum	Maximum	Mean	Standard Deviation	Minimum	Maximum
1	1	1	Vector	11.073	0.655	8.974	13.255	427	314	16	2014
1	2	2	Vector	10.123	0.509	8.388	11.386	459	293	17	1609
1	2	3	Vector	10.015	0.497	8.388	11.762	478	376	16	2277
1	3	3	Vector	9.527	0.356	8.664	10.474	482	355	16	2043
2	4	4	Vector	8.877	0.264	8.388	9.598	541	316	41	1551
1	5	1	Vector	9.191	0.313	8.388	9.999	461	323	16	1517
1	1	1	Multiplication	8.394	0.041	8.388	8.848	107	29	41	290
1	2	1	Multiplication	8.389	0.016	8.388	8.664	71	19	31	138
1	2	2	Multiplication	8.388	0.000	8.388	8.388	43	6	24	64

Table 7: Valley terrain results

As the importance of the Visibility and Cost metrics is increased the results obtained are also better in the sense that the mean path energy is less. However this is a purely greedy algorithm trait. The exploration characteristics of the algorithm are severely hindered as can be seen by the number of iterations taken to reach a solution and also from qualitative observations made during run time.

On first observation achieving the optimum solution 100% of the time may appear to be desirable, however it is actually a weakness in the search characteristics of the algorithm, which if unchecked could lead to premature convergence on sub-optimal solutions in some data sets.

Over the range of parameter values used for the multiplicative combination technique the optimum solution was found at least 90% of the time in 1000 runs, and even if the optimum solution was not found, a solution which lies very close to optimum was found instead.

The vector combination method did not perform as well as the multiplication combination method. This may have been a result of poor choice of parameter values that lead to the vector technique spending too much time exploring the terrain rather than exploiting good paths.

6.4 Symmetries in terrain data sets

Due to the start and finish vertices of each data set being at the same elevation an interesting property exists, that is that all data sets can be solved in reverse to obtain the same optimum total path energy from end to start.

Experiments were conducted in which 50% of the ants were initialised at the end vertex rather than 100% of the colony being initialised at the start vertex to determine the effect this resource allocation would have on the way the ants interacted in the environment.

For the mound terrain the optimum path was still found in 100% of runs however the mean number of iterations for this solution doubled. This is not unexpected as the number of ants solving the conventional optimum path (start to end vertex) has been halved, half of the resources are now consumed in the task of finding an optimum path from end to start.

For the volcano terrain the optimum path was found in only 75% of trials using the same parameters as before. As with the mound terrain the time to solution was doubled due to the resources being split across the problem. The effect of adding ants at the end vertex was a reduction in the performance of the algorithm for this terrain.

When ants were initialised at the end vertex as well as the start vertex the results for the Valley terrain were far worse than what had been achieved previously using the same parameters. Table 8 shows how the mean iterations was in some cases 4 times larger than previous results, and that the solution quality (path energy) was also worse than the previous section.

Problem Parameters				Path Energy				Iterations			
Alpha (Phe)	Beta (Vis)	Gamma (Cost)	Combination Method	Mean	Standard Deviation	Minimum	Maximum	Mean	Standard Deviation	Minimum	Maximum
1	1	1	Vector	12.002	0.887	9.230	14.866	378	309	17	1990
1	2	2	Vector	11.332	0.734	9.123	13.431	384	313	17	2089
1	2	3	Vector	11.304	0.720	8.848	13.203	366	301	16	1590
1	3	3	Vector	10.792	0.613	8.698	12.375	381	310	16	2042
2	4	4	Vector	10.851	0.650	8.664	13.450	398	321	16	1889
1	5	1	Vector	10.068	0.501	8.388	11.558	381	308	16	1605
1	1	1	Multiplication	9.509	0.680	8.388	12.155	509	230	16	1354
1	2	1	Multiplication	8.705	0.307	8.388	9.849	380	174	39	1105
1	2	2	Multiplication	8.397	0.056	8.388	8.954	176	77	46	842
1	3	3	Multiplication	8.388	0.000	8.388	8.388	48	9	16	88

Table 8: Valley terrain results with ants initialised at end and start vertices

Since the terrain sets are asymmetric the ants seem to interfere with each other rather than assist each other in creating composite solutions. Initialising ants at the start and end vertex may prove more effective if the terrain set was symmetric, this way the direction of travel would not interfere with the creation of a solution. It is concluded however that when solving an asymmetric problem it is better to initialise the entire population at the start vertex rather than split the population between the start and end.

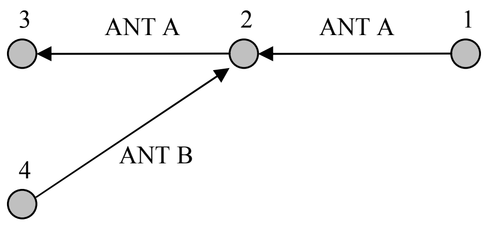


Figure 13: Example of pheromone interference

Pheromone trails are direction dependent as edge (i,j) ≠ edge (j,i). An example of interference can be seen in Figure 13. Ant A has laid pheromone on the path from vertex 1 to 2 to 3. Ant B, on reaching vertex 2 is not attracted by the pheromone from 2 to 1 (ant A has laid pheromone from 1 to 2), however it is attracted by the pheromone from vertex 2 to 3, which while good for ant A may not be good for ant B. This will effectively direct ant B towards ant A's target vertex which is opposite to ant A's target vertex.

7 Project Extension

A problem that may prove to be worthy of further research is a combined TSP/Shortest Path problem:

Given a set of n connected vertices find a roundtrip tour of shortest weight that must include m major vertices (once only), where m є n.

Suggested techniques for solving such a problem include the division of the problem into a series of shortest path problems the results of which could be used in a conventional TSP. This approach if implemented using efficient shortest path algorithms could prove effective, however the degree to which it would scale as the dimensionality increased is unknown.

A seemingly more intuitive approach may be to attempt to tackle this as one problem solving both the TSP and shortest paths in a combined fashion.

A difficulty in approaching this as a combined problem is what happens to the visibility metric? The current visibility metric has been optimised for a single major vertex, however there now exists more than one possible target vertex out of the subset of major vertices, and there exists no priori knowledge as to the best target vertex to aim for.

There exists currently three possibilities for this problem:

- Let the ant decide its next target vertex upon leaving a major vertex.
- Let the ant alter its target vertex as it travels but at any time be concentrating on only one.
- Use a net visibility value that is proportional to the sum of the visibility of all remaining target vertices.

At this stage it is unknown which approach is best, or if indeed some combination may prove viable.

8 Conclusion

In this initial investigation ACO has been shown to be suitable for use in solving the data sets constructed. The identified weakness of the algorithm is its dependence on the correct implementation of the pheromone metric.

Initially it is desirable for the ants to explore the problem space quite vigorously, whereas as time progresses it is preferred that the ants to converge upon a selected solution. The pheromone implementation used in this investigation was quite sensitive to small changes and hence it was often found that some sub-optimal path's pheromone level became overly enhanced quite early during the 'exploration' phase. This lead to a premature convergence on sub-optimal solutions sometimes and made the algorithm somewhat unpredictable.

A possible solution to the problem above would be to make the pheromone importance dynamic, in the sense that less attention is placed on the metric initially, and after sufficient runs the importance is slowly increased until convergence on a solution.

By initialising the pheromone to zero and using the vector combination approach this will automatically. Unfortunately by the time this incompatibility of the vector combination with non-zero pheromone initialisation was identified it was not possible to conduct experiments with zero initial pheromone in the time remaining. This would be a desirable extension to prepare this work for publication.

As it is the algorithm has no problem converging upon a solution, and it often makes small changes to a path to align it more closely with the optimum path. With further refinement made to the pheromone metric to ensure initial exploration, the SPACO algorithm should be able to confidently solve more challenging data sets, challenging in both increased size and increased complexity.

The speed with which the algorithm is able to find a solution compared to a traditional exhaustive search technique such as a depth-first search or breadth-first search is exceptional, with solution quality only slightly degraded. The results obtained in Section 6.3 show how the SPACO algorithm can also be modified for either increased exploration or exploitation of the problem space.

9 References

1 Dorigo, M. (1992) "Optimization, Learning and Natural Algorithms", PhD Thesis, Dipartimento di Elettronica, Politechico di Milano, Italy.

2 Dorigo, M. and Gambardella, L. (1997) "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", IEEE Transactions on Evolutionary Computing, 1, pp. 53-66.

3 Dorigo, M and Gambardella, L. (1997) "Ant Colonies for the Traveling Salesman Problem", Biosystems, 43, pp. 73-81.

4 Dorigo, M. and Di Caro, G. (1999) "The Ant Colony Optimization Meta-heuristic", in New Ideas in Optimization, Corne, D., Dorigo, M. and Golver, F. (eds), McGraw-Hill, pp. 11-32.

5 Dorigo, M., Maniezzo, V. and Colorni, A. (1996) "The Ant System: Optimization by a Colony of Cooperating Agents", IEEE Transactions on Systems, Man and Cybernetics - Part B, 26, pp. 29-41.

6 Glover, F. and Laguna, M. (1997) "Tabu Search", Kluwer Academic Publishers, Boston: MA, 442 pages.

7 Goldberg, D. (1989) "Genetic Algorithms in Search, Optimization and Machine Learning", Addison Wesley: Reading, MA, 412 pages.

8 Reinelt, G. TSPLIB95. Available from http://www.iwr.uniheidelberg.de/iwr/comopt /soft /TSPLIB95/TSPLIB95.html

9 St¨utzle, T. and Dorigo, M. (1999) "ACO Algorithms for the Traveling Salesman Problem", in Evolutionary Algorithms in Engineering and Computer Science, Miettinen, K., Makela, M., Neittaanmaki, P. and Periaux, J. (eds), Wiley.

10 van Laarhoven, L. and Aarts, E. (1987) "Simulated Annealing: Theory and Applications", D Reidel Publishing Company: Dordecht, 186 pages.

11 E.W. Dijkstra. (1959) "A note on two problems in connexion with graphs", Numerische Matematik, 1:269--271.

12 Bellman, R. (1957) "Dynamic Programming", Princeton University Press, Princeton, New Jersey.

10 Appendices

10.1 Problem Parameters

10.1.1 Flat terrain

Number of ants	9
Total System Pheromone (total distributed evenly across all edges)	40
Pheromone decay	5%
Pheromone update	0.2222
Pheromone Importance (Power)	1
Visibility Importance (Power)	2
Cost Importance (Power)	1

10.1.2 Mound terrain

Number of ants	25
Total System Pheromone (total distributed evenly across all edges)	144
Pheromone decay	5%
Pheromone update	0.288
Pheromone Importance (Power)	1
Visibility Importance (Power)	3
Cost Importance (Power)	1

10.1.3 Volcano terrain

Number of ants	81
Total System Pheromone (total distributed evenly across all edges)	544
Pheromone decay	5%
Pheromone update	0.336
Pheromone Importance (Power)	1
Visibility Importance (Power)	2
Cost Importance (Power)	2

10.1.4 Valley terrain

Number of ants	289
Total System Pheromone (total distributed evenly across all edges)	2112
Pheromone decay	5%
Pheromone update	0.365