International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:2, 2008

Grid Based and Random Based Ant Colony Algorithms for Automatic Hose Routing in 3D Space

Gishantha Thantulage, Tatiana Kalganova, and Manissa Wilson

***Abstract*—**Ant Colony Algorithms have been applied to difficult combinatorial optimization problems such as the travelling salesman problem and the quadratic assignment problem. In this paper grid-based and random-based ant colony algorithms are proposed for automatic 3D hose routing and their pros and cons are discussed. The algorithm uses the tessellated format for the obstacles and the generated hoses in order to detect collisions. The representation of obstacles and hoses in the tessellated format greatly helps the algorithm towards handling free-form objects and speeds up computation. The performance of algorithm has been tested on a number of 3D models.

***Keywords*—**Ant colony algorithm, Automatic hose routing, tessellated format, RAPID.

I. Introduction

HOSE and harness routing is a significant research area in assembly design. Many CAD and solid model manufacturers incorporate the ability to represent these components in their products. However, the programs available are not always able to produce efficient routing. Often, skilled personnel who understand the engineering requirements, the model representations and physical production issues fill this technical gap. This requires human intervention to create assemblies and as CAD design tools allow rapid design and redesign of products at speeds that exceed the current human capacity, hose and harness routing cannot be done efficiently. There is an unacceptable bottleneck in meeting the customer's demand when bringing products to the market.

Most hose routing problems are difficult combinatorial optimization problems and combinatorial optimization techniques such as genetic algorithms, ant colony algorithms and simulated annealing can be used to produce a feasible set close to the optimal solution.

Manuscript received May 31, 2006. This work was supported in part by the EPSRC.

Gishantha Thantulage is with the Bio Inspired Intelligent System Group (BIIS), School of Engineering and Design, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK (phone: 0044-77-1013-1968; fax: 0044-1895-258728; e-mail: gishantha@ ieee.org).

Tatiana Kalganova and Manissa Wilson are with the School of Engineering and Design, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK (phone 0044 1895 274 000; fax 0044-1895-258728, e-mail: Tatiana.Kalganova@brunel.ac.uk; Manissa.Wilson@brunel.ac.uk).

In the initial stage of this research, hose routing can briefly be defined as finding a collision free path (optimal or near optimal) between a start point and a target point.

In this paper, two ways of representing the road map for the ant colony algorithm are presented; one is a fixed-sized grid map and the other consists of randomly selecting points from the world. To handle collision detection and achieve CAD software independency, the algorithm uses the tessellated format for the obstacles (which is available in most CAD and computer graphics packages) and the collision detection library RAPID.

The structure of the algorithm is summarized in Fig. 1.

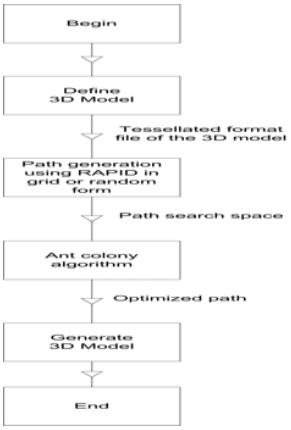


Fig. 1 Structure of the ant colony algorithm for automatic hose routing

The rest of the paper is structured as follows. Section II describes the ant colony algorithm. Section III provides a description of the tessellated format (or representation) of the CAD models and a description of the collision detection library RAPID. Section IV describes the implementation of the two versions of the ant colony algorithm. Section V presents the simulation results for the two versions. The results are discussed in Section VI and Section VII concludes the paper.

II. Ant Colony Algorithm

Ant colony algorithms were first proposed by Dorigo and

his colleagues [1], [4] as a multi-agent approach to difficult combinatorial problems such as the travelling salesman problem and the quadratic assignment problem. Later scientists have applied them to many different discrete optimization problems summarized in [2], [3] and [6]. In this paper, the ant colony algorithm is applied to 3D hose routing in assemblies using two types of road maps.

Real ants are able to find the shortest path between their nest and a food source. Communication between the ants is based on a pheromone trail deposited by individual ants. An ant's tendency to choose a specific path depends on the intensity of the pheromone trail on the path, i.e., the stronger the pheromone trail a path has the higher the probability that an ant will follow that particular path. Over time, the pheromone trail evaporates and it loses intensity if no more pheromone is laid down by other ants. If a large number of ants choose a specific path, the intensity of this trail increases and more ants tend to choose that path.

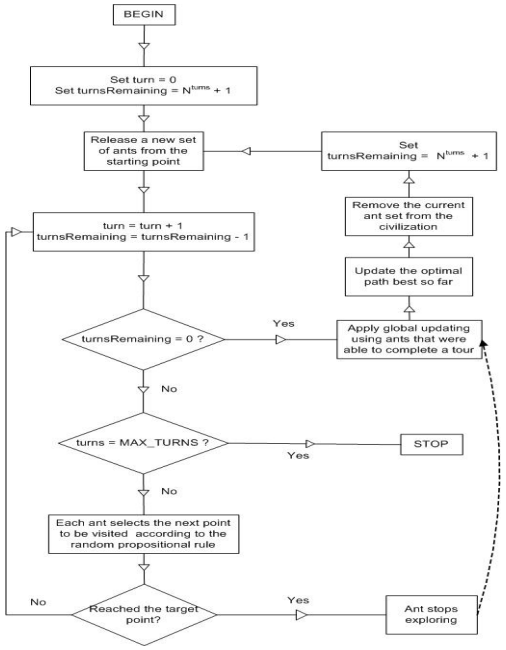


Fig. 2 Flowchart of the ant colony algorithm

Ants perform a complete tour (in this paper a tour is defined as travelling from the start point to the target point) by choosing grid points or random points according to a probabilistic state transition rule which selects neighbouring points that are closest to the target point and have a high amount of pheromone. Once all the ants have completed a certain number of turns (N^{turns}) a global pheromone updating rule (global updating rule, for short) is applied (see Fig. 2). A fraction of the pheromone evaporates on all edges (edges that are not refreshed become less desirable); each ant that was able to finish a complete tour, deposits an amount of pheromone on the edges which belong to its tour in proportion to how short its tour was (in other words, edges which belong to many short tours receive the greater amount of pheromone). After global updating, the current set of ants is removed from the civilization and another set of ants starts from the start point to explore the target point. The process is iterated until the number of turns reaches a maximum number of turns (*MAX_TURNS*). Note that, the parameter N^{turns} is set such that most of the ants in the initial set are able to reach the target point.

The state transition rule used by ant the system, called a random-proportional rule, is given by (1) and gives the probability with which ant k in city r chooses to move to city s [5],

$$p_k(r,s) = \begin{cases} \dfrac{[\tau(r,s)].[\eta(r,s)]^{\beta}}{\sum_{u \in J_k(r)} [\tau(r,u)].[\eta(r,u)]^{\beta}}, & if \ \ s \in J_k(r) \\ 0, & otherwise \end{cases} \quad (1)$$

where τ is the pheromone level, $\eta = 1/\delta$ is the inverse of the distance (δ) from point s to the target point, $J_k(r)$ is the set of neighbour points of r that remain to be visited by ant k positioned on the point r (to make the solution feasible), and β is a parameter which determines the relative importance of pheromone versus distance ($\beta > 0$).

In the ant system, the global updating rule is implemented as follows. Ants those were able to complete their tour within the number of allocated turns (N^{turns}), allow updating pheromone levels of their visited edges according to [5],

$$\tau(r,s) \leftarrow (1-\rho).\tau(r,s) + \sum_{k=1}^{m} \Delta\tau_k(r,s) \quad (2)$$

where

$$\Delta\tau_k(r,s) = \begin{cases} \dfrac{1}{L_k}, & if\,(r,s) \in tour\ done\ by\ ant\ k \\ 0, & otherwise \end{cases} \quad (3)$$

$0 < \rho < 1$ is a pheromone decay parameter, L_k is the length of the tour performed by ant k, and m is the number of ants that were able to complete their tour within the stipulated number of turns N^{turns}.

III. The Tessellated Format and RAPID

The search space is represented by a 3D model. The 3D model can be generated by using any CAD software such as Pro/Engineer or AutoCAD. Most CAD software supports the tessellated representation of a 3D model. Therefore, in order to provide a CAD software independent implementation of the algorithm, the 3D model was represented in the tessellated format.

The .stl (STereoLithography) format [7] is an ASCII or binary file used in manufacturing. It is a list of triangular planes that describes a computer generated solid model. This is the standard input for most RAPID prototyping machines. A .stl file defines an object's surfaces as a set of adjacent triangles as shown in Fig. 3. The file basically contains X, Y and Z Cartesian coordinates of each vertex of the triangles, as well as the coordinates of the vectors normal to the triangles. With the tessellated format, each edge is shared only by two triangles. The tessellated model is an approximation to the real

model and the accuracy of the tessellated model depends on the number of triangles used. In most CAD packages the number of triangles generated for the tessellated model can be controlled. Models were generated using the CAD package Pro/Engineer and its programming toolkit Pro/Toolkit.

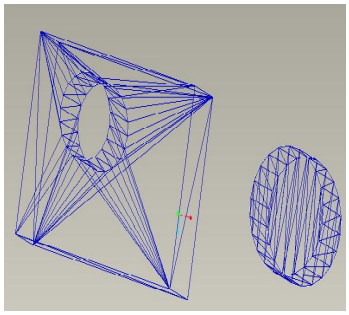


Fig. 3 Tessellated representation of objects

The proposed algorithm is based on the identification of available paths in the given 3D model represented by .stl format. The availability of paths can be determined by the collision detection library RAPID.

RAPID (Robust and Accurate Polygon Interface Detection) [8] is a C++ library developed at Department of Computer Science, University of North Carolina, for interference detection (or collision detection) of large environments composed of unstructured models.

- It is applicable to polygon soups [8] - models that contain no adjacency information and obey no topological constraints. The models may contain cracks, holes, self-intersections, and non-generic (e.g. coplanar and collinear) configurations.
- It is numerically robust - the algorithm is not subject to conditioning problems and requires no special handling of non-generic cases (such as parallel faces).
- The RAPID, library is free for non-commercial use. It has a very simple user interface: the user needs to be familiar with only about five function calls.

RAPID accepts only polygonal models composed entirely of triangles, but does not require the model to have any particular structure. For example, some collision detection systems require the shapes to be well-formed solids – the surfaces must be "closed" so that there are a well-defined inside and outside.

IV. Implementation

The algorithm was implemented in three steps. In the first step, the tessellated representation of the obstacles was obtained as a text file from the CAD package. This file was passed to a program which incorporated the collision detection library RAPID. The following inputs were also supplied to the program:

- world Size of the paths to be explored, given by the maximum and minimum of each coordinate axis - X^{min}, X^{max}, Y^{min}, Y^{max}, Z^{min}, and Z^{max},
- coordinates of the start point $S(X_S, Y_S, Z_S)$ and target point $T(X_T, Y_T, Z_T)$,
- number of ants to be released,
- values for the parameters ρ (pheromone decay parameter) and β,
- initial pheromone levels of the edges (constant),
- number of turns for which the algorithm is to be run (*MAX_TURNS*),
- frequency at which the global pheromone update rule is applied (N^{turns}),
- radius (r) of the hose or pipe segment.

It was not possible to find a benchmark for a comparison study with previous work on pipe routing. Automatic pipe routing has previously been addressed in [7]; the authors used genetic algorithms and RAPID for pipe routing and applied them only to one real-world application and took hours to obtain the optimal path or near optimal path. Therefore, at the initial stage, the implementation of the algorithm was restricted to models specifically generated for the experiments. Furthermore, the main goal was to conduct a feasibility study of applying the ant colony algorithm for automatic 3D hose/pipe routing. In future work, the algorithm will be applied to some real-world applications.

In the second step, the program implemented three tasks.

Firstly, it created the whole road map for the two versions, rectangular grid or randomly generated points from the world. When connecting two points, the program checked, with aid of the C++ library, RAPID, that the path between the two points was collision free (the axis of the hose cylinder lies on the line connecting the two points). For simplicity, a rectangular hexahedron was used that was centred on the line segment between the two points such that the cylindrical hose could be laid within it. For the first version (fixed-size grid map), when trying to connect a grid point to another, the algorithm considered only north, west, south, east, top and bottom neighbours (6-way connection) as this would reduce the number of routes needed to be stored in the memory. The road map was stored in a text file that can be used again if the algorithm needs to be executed another time. For the second version, the program generated points randomly from the world. Then as for the first version, the algorithm tried to connect each and every point (including the start and target points) within the randomly generated point set.

Secondly, the program searched for optimal path or near optimal path between the start and the target points using the ant colony algorithm and the roads maps created earlier. If paths contained cycles, these were removed before applying global updating of the pheromone. Initially, a constant pheromone value was set for each edge. Before applying global updating, the program found the optimal path for the current set of ants and if this was an improvement on the path for the previous set of ants, it set this path as the optimum path found so far.

Thirdly, at the end of *MAX_TURNS*, the (optimal) path obtained for the grid-based version was further refined to eliminate some 'staircases' (see Fig. 4). Again, when refining

the optimal path, before connecting two points, the algorithm used RAPID to detect any collisions.

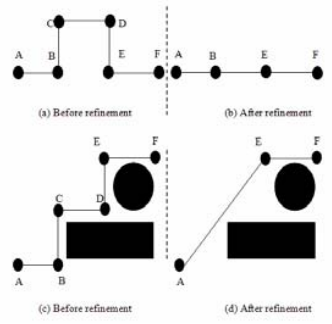


Fig. 4 Refining the path

In the third step, the program generated the list of points needed for moving from the start point to the target point.

V. Simulations

Two versions of the ant colony algorithm were implemented and their strengths and weaknesses were investigated experimentally. The CAD Pro/Engineer package was used for generating the 3D models and its programming toolkit Pro/Toolkit, was used for obtaining the tessellated format of the generated models.

The parameter settings for the ant colony algorithm were: number of ants = 10, initial pheromone level for each edge = 100, number of turns for which the algorithm is to be run, *MAX_TURNS* = 10,000, pheromone decay parameter ρ = 0.01, and $\beta = 5$.

All the simulations were conducted on a Pentium IV PC (Processor speed = 3.0 GHz, Memory = 512 MB) in the Microsoft Windows XP environment using Microsoft Visual C++ (Version 6.0).

The performance of the algorithm was defined by time (seconds) and the length of the optimal path.

For each model, the grid-based version was tested on 3 different step sizes (increment values of the x, y and z coordinates) 10, 25, and 50. The random-based version was tested for 100, 150, 200 and 500 random points. All the simulations were carried out for 10,000 turns and averaged over 10 trials.

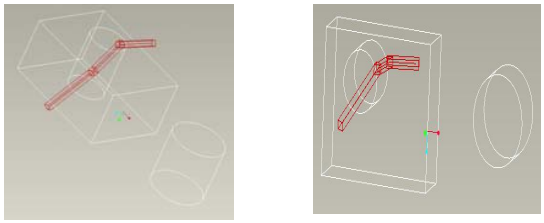
In the figures below, the best paths obtained over 10 trials are shown for the two versions.

A. Model 1: Hose Routing in an Environment with a Hole in a Cube

TABLE I
Comparison of Grid-Based and Random-Based - Hole in a Cube Models

	Grid-based			Random-based			
Step Size	10	25	50	N/A	N/A	N/A	N/A
No of points	18081	1377	225	100	150	200	500
Avg. Length	**244.32**	251.66	378.80	292.70	299.00	262.49	**245.22**
St. Dev. (Len.)	4.21	4.35	11.24	51.20	54.00	17.51	3.40
Best (Length)	**237.79**	247.23	367.96	256.80	257.70	241.32	**238.85**
Avg. Time (s)	993.70	49.40	3.50	8.90	21.40	32.00	220.10

The proposed ant colony algorithm was tested in an environment consisting of a cube containing a hole (see Fig. 5). Hose segments needed to be laid inside this hole in order to obtain the optimal path.



(a) Grid-based (b) Random-based

Fig. 5 Model 1: Hole in a cube
{X^{min} = -250, X^{max} = 150, Y^{min} = -50, Y^{max} = 150, Z^{min} = -200, Z^{max} = 0; S = (-200, 150, -100); T = (-100, -50, -150);
Radius = 5; N^{turns} = 100}

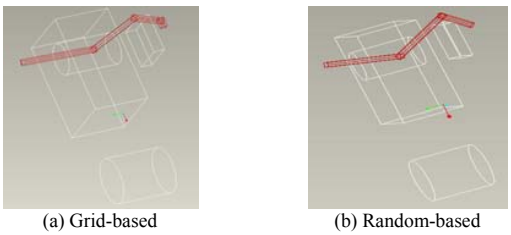
Table I shows the comparison of the two versions over 10 trials for each value of the step size (version 1) and each number of random points (version 2). According to Table I, the optimal solution generated by the random-based version (with 500 random points) is very close to the optimal solution generated by the grid-based version (with step size = 10 and 18081 points). However, the average computational time taken by the random-based version is comparatively less than for the grid-based version (220.1 sec. against 993.7 sec.). The random-based version is more than 4 times faster.

B. Hose Routing in an Environment with a Hole in a Cube where the Optimal Path is Blocked by an Obstacle

In this simulation, the optimal path found in the earlier case was blocked by a cubic obstacle and the target point was placed behind the obstacle (see Fig. 6 and Table II).

TABLE II
Comparison of Grid-Based and Random-Based - Hole in a Cube Models where the Optimal Path is Blocked by a Cubic Obstacle

	Grid-based			Random-based			
Step Size	10	25	50	N/A	N/A	N/A	N/A
No of points	18081	1377	225	100	150	200	500
Avg. Length	322.70	**305.08**	400.00	397.70	368.10	376.50	**312.60**
St. Dev. (Len.)	20.96	7.24	0.00	43.10	29.88	41.70	15.14
Best (Length)	**290.55**	299.34	400.00	339.40	329.54	324.40	**291.51**
Avg. Time (s)	952.20	64.40	5.00	12.30	27.30	49.90	318.30



(a) Grid-based (b) Random-based

Fig. 6 Model 2: Optimal path is blocked by a cubic obstacle
{X^{min} = -250, X^{max} = 150, Y^{min} = -50, Y^{max} = 150, Z^{min} = -200, Z^{max} = 0; S = (-200, 150, -100); T = (-200, -100, -150);
Radius = 5; N^{turns} = 100}

The best average length for the grid-based version is

obtained with step size 25 (points 1377). However, the best path was produced with the step size 10. The average path length of the random-based version with 500 points (312.6) is relatively close to the average path length of the grid-based version with step size 25 (305.08) and the lengths of the best paths for in the both versions are very close.

This experiment shows that when selecting the right grid (or step) size, the grid-based version performs very well even with relatively large step sizes.

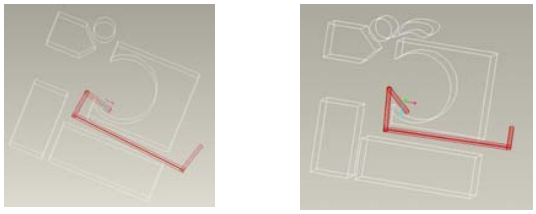
C. Hose Routing in an Environment with a U-Shape Obstacles

In this experiment, a U-shape obstacle was placed in the environment and the environment was made more complex by introducing other objects. Furthermore, the start and the target points were placed such that only one path existed between them. Note that the z coordinates of the search space were restricted to the top and the bottom of the obstacles (See Fig. 7 and Table III).

In this experiment, the grid-based version failed in all the trials with the step sizes 10 and 50. However, it was successful with step size 25 and generated the best average length (596.28) and best optimal path lengths (551.31). This experiment demonstrates that if the right resolution is selected, the grid-based version performs well in terms of both optimal length and the computational time.

TABLE III
COMPARISON OF GRID-BASED AND RANDOM-BASED MODELS WITH A U-SHAPED OBSTACLE

	Grid-based			Random-based			
Step Size	10	25	50	N/A	N/A	N/A	N/A
No of points	55451	4205	675	100	150	200	500
Avg. Length	Failed	**596.28**	Failed	1260.00	925.00	761.60	**710.30**
St. Dev. (Len.)		22.42		405.00	328.00	174.80	40.00
Best (Length)		**551.31**		698.00	654	619.10	**608.9**
Avg. Time (s)		173.25		13.20	38.10	89.70	804.60



(a) Grid-based (b) Random-based

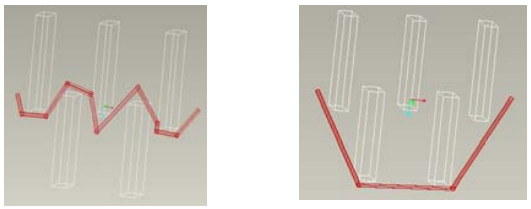
Fig. 7 Model 3: U-shaped obstacle
{X^{min} = -300, X^{max} = 400, Y^{min} = 0, Y^{max} = 100, Z^{min} = -300, Z^{max} = 400; S = (50, 25, -50); T = (350, 25, -50); Radius = 5; N^{turns} = 100}

D. Hose Routing in an Environment with Parallel Walls

In this experiment, two 3D points were selected and the shortest path between them was blocked by 5 parallel walls (see Fig. 8 and Table IV).

TABLE IV
COMPARISON OF GRID-BASED AND RANDOM-BASED MODELS CONTAINING PARALLEL WALLS

	Grid-based			Random-based			
Step Size	10	25	50	N/A	N/A	N/A	N/A
No of points	40931	3125	507	100	150	200	500
Avg. Length	Failed	**1096.90**	Failed	1021.90	1025.40	986.20	**963.61**
St. Dev. (Len.)		91.30		41.60	57.20	45.90	9.07
Best (Length)		**1007.70**		968.90	963.00	**938.00**	948.09
Avg. Time (s)		133.67		11.10	28.30	59.40	316.70



(a) Grid-based (b) Random-based

Fig. 8 Model 4: Parallel walls {X^{min} = -300, X^{max} = 300, Y^{min} = 0, Y^{max} = 100, Z^{min} = -300, Z^{max} = 300; S = (-300, 25, 0); T = (300, 50, -25); Radius = 5; N^{turns} = 100}

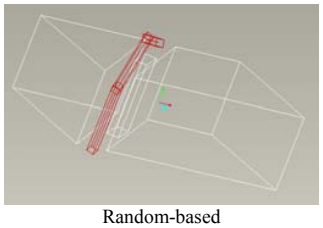
Here also, the grid-based version failed for step sizes 10 and 50. Even though, the grid-based version was successful with step size 25, the average length and the best length are higher than the respective values for the random-based version. The average computational time for the random-based version is low for all cases except for 500 random points.

E. Hose Routing in an Environment with a Diagonal Empty Space

In this simulation, a diagonal empty space was placed between two objects and the straight path between the start point and the target point was blocked by a cubic shaped object (see Fig. 9 and Table V). The grid-based version failed in all 3 cases. The random-based version was successful in each case and produced reasonable results.

TABLE V
COMPARISON OF GRID-BASED AND RANDOM-BASED MODELS WITH A DIAGONAL EMPTY SPACE

	Grid-based			Random-based			
Step Size	10	25	50	N/A	N/A	N/A	N/A
No of points	22386	1683	270	100	150	200	500
Avg. Length	Failed	Failed	Failed	320.24	313.32	327.17	**295.23**
St. Dev. (Len.)				13.70	9.21	20.44	9.86
Best (Length)				302.38	299.14	303.08	**275.39**
Avg. Time (s)				7.60	16.60	27.50	155.70



Random-based

Fig. 9 Model 5: Diagonal empty space
{X^{min} = -50, X^{max} = 350, Y^{min} = -50, Y^{max} = 150, Z^{min} = -200, Z^{max} = 50; S = (25, 0, 0); T = (180, 100, -185); Radius = 5; N^{turns} = 100}

VI. Discussion

Previously, scientists have applied the ant colony algorithm to many real-world problems such as the travelling salesman problem (TSP), the quadratic assignment problem, and job shop scheduling. In this paper, it has been applied to automatic 3D hose/pipe routing where the world is represented as two versions, grid-based and random-based.

The problem presented in this paper and the TSP is quite similar; however there are some differences. In the TSP, paths must be found such that each ant must travel to each city once and must finally come back to the start city. In the case described in this paper, ants must start from the start point and need to finally reach the target point. The constraints that each ant must travel to each point and that ants must finally come back to the start point are not imposed. However, it must be guaranteed that when an ant has visited a point, it must not visit that point again. To this end, cycles were removed from the ants' paths before applying the global updating rule. For the TSP, the global updating rule is applied after all ants have completed a tour (i.e., each and every ant must come back to the start city). Hence, for the TSP, the algorithm knows when to apply the global updating rule. In the experiments described above, this is not always possible, as some ants may get lost. Thus, a new parameter, N^{turns}, was introduced into the algorithm. This parameter was set such that most of the ants of the current set were able to reach the target point.

The above simulation results show the strengths and weaknesses of the grid-based and the random-based of the ant colony algorithm for automatic 3D hose routing. The simulation results show that both versions can be applied for any shape which can be generated using any CAD software. The use of the RAPID library greatly helps the algorithm to detect collisions when laying the hoses.

The simulation study also indicates that the proposed grid-based and random-based versions of the ant colony algorithms are of practical use because the required computational times are reasonably low.

However, in the grid-based version the resolution or the size of the grid plays an important role in the determination of the optimal path and affects the computational time. If none of the grid line falls on the optimal path when constructing the road map, the algorithm fails to obtain the optimal path (See Tables III, IV and V). Thus, selecting the right size of grid (or step size) is important for the grid-based version.

The advantage of using the random-based version is that it did not fail for any of the tested models and produced a reasonably good solution to the problem. Furthermore, in the case of grid-based version, as the step size is decreased, amount of memory needed to store the road map increases drastically as does computation time.

VII. Conclusion

In this paper, two versions of ant colony algorithm, namely grid-based and random-based, have been proposed for automatic 3D hose routing. For both versions the algorithm generates the optimal set of pipe segments linking the start and the target points. The C++ library, RAPID, is incorporated into the program for collision detections. The .stl format of the obstacles is passed to the algorithm as the RAPID can only handle triangular shapes. However, the accuracy of the collision detection depends on the number of triangles used to approximate the obstacles. The effectiveness of both versions of the algorithm is demonstrated by simulation studies. The simulation results shows that proposed algorithm can handle complex environments and any shape that can be generated using any CAD software. The computational efficiency suggests that the algorithm can be applied to real-world hose/pipe routing problems.

The selection of the right resolution (or step size of the grid) of the grid-based version plays an important part and is dependent on the problem at hand (See Tables III, IV and V). When the resolution increased, the algorithm requires higher amounts of memory and more time to compute the results. However, if the correct resolution is selected, the grid-based version can in some cases provide the best solution

In the random-based version, the algorithm was able to find a reasonably good solution in a reasonable time for any of the tested models.

At this initial stage of the research, the algorithm has been implemented only for optimizing the distance between the start and target points. In the next stage, other hose routing knowledge will be incorporated into the algorithm, such as, the selection of pipe bends from a pre-specified catalogue of angles of bends, the minimizing of the cost of pipes, the avoidance of hot, sensitive and moving objects. Other combinatorial optimization algorithms will also be implemented, such as genetic algorithms and quantum-inspired genetic algorithms for automatic 3D hose routing and these will be compared to the results with the ant colony algorithm presented here.

Acknowledgment

The authors acknowledge BIIS research group at Brunel University, UK for providing valuable comments during the course of this research work.

References

[1] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, Vol. 26, No. 1, pp. 1-13.

[2] Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). From natural to artificial swarm intelligence. New York: Oxford University Press.

[3] Corne, D. Dorigo, M., & Glover, F. (Eds.). (1999). New ideas in optimization. Maidnhead, UK: McGraw-Hill.

[4] Gambardella, L.M., & Dorigo, M. (1996). Solving symmetric and Asymmetric TSPs by ant colonies. Proceedings of IEEE International Conference. pp. 622-627.

[5] Gambardella, L.M., & Dorigo, M. (1997). Ant Colony System: A cooperative learning approach to the travelling salesman problem. Evolutionary Computation, IEEE Transactions. pp. 53-66.

[6] Gambardella, L.M., & Dorigo, M. (1999). Ant algorithms for discrete optimization. Artificial Life 5: Massachusetts Institute of Technology. pp. 137-172.

[7] Sandurkar, S., & Chen, W. (1998). GAPRUS – Genetic algorithms based pipe routing using tessellated objects. The journal of computers in industry.
[8] Gottschalk, S., Lin, M.C., & Manocha, D. RAPID (Robust and Accurate Polygon Interface Detection). Available: http://www.cs.unc.edu/~geom/OBB/OBBT.html