2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

A Grid-based Ant Colony Algorithm for Automatic 3D Hose Routing

Gishantha Thantulage, *Member, IEEE,* Tatiana Kalganova, W.A.C. Fernando, *Member, IEEE*

***Abstract*— Ant Colony Algorithms applied to difficult combinatorial optimization problems such as the traveling salesman problem (TSP) and the quadratic assignment problem. In this paper we propose a grid-based ant colony algorithm for automatic 3D hose routing. Algorithm uses the tessellated format of 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 speed up the computations. The performance of the algorithm has been tested on a number of 3D models.**

I. Introduction

PATH finding is an important problem for many applications, including network traffic, robotic planning, military simulations, computer games, vehicle routing, electric circuit routing and hose routing. Path finding involves analyzing a road map to find the best cost of travelling from one point to another. Best can be a multi-objective function and use such criteria as the shortest path, least-cost path, safest path, etc.

Hose routing is a major research area in assembly design. Almost all mechanical assemblies include pipes, cables and hoses. Hose routing can be briefly defined as finding a collision free path between the start and target points.

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

In this paper, we present a grid based ant-colony algorithm for automatic 3D hose routing. Algorithm uses tessellated representation of the obstacles (which is available for most of the CAD and computer graphics packages) for collision detection and hence eliminates the restrictions on shapes of the obstacles and their representations.

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

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 description of the collision detection library RAPID. Section IV describes the implementation of the grid-based ant colony algorithm. Section V presents the simulation results of the algorithm. The results are discussed in section VI and section VII concludes paper.

This work was supported in part by the EPSRC.

The authors are with the Bio Inspired Intelligent System Group (BIIS), School of Engineering and Design, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK (gishantha@ieee.org).

II. The Ant System

Ant algorithms were first proposed by Dorigo and his colleagues [1, 4] as multi-agent approach to difficult combinatorial problems such as the travelling salesman problem (TSP) and the quadratic assignment problem. Later scientists apply them to many different discrete optimization problems summarized in [2, 3, 6]. In this paper, we apply ant system for 3D hose routing in assemblies.

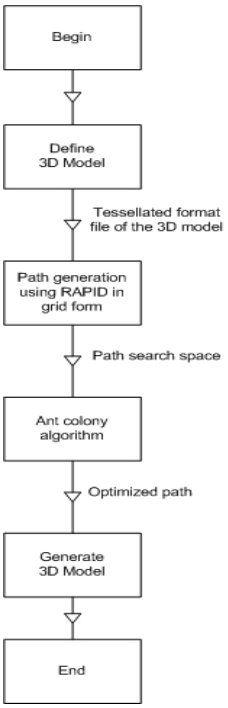


Fig. 1 Structure of the grid-based ant colony algorithm for automatic hose routing

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

0-7803-9487-9/06/$20.00/©2006 IEEE

ants choose a specific path, the intensity of this trail increases and more ants tend to choose that path.

Ants perform a complete tour (in our case tour is defined as travelling from start point to the target point) by choosing the grid points according to a probabilistic state transition rule (random-proportional rule) which selects neighbouring points that are closest to the target point and have a high amount pheromone. Once all ants have completed 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 who were able to finish a complete tour, deposits an amount of pheromone on edges which belong to its tour in proportion to how short its tour was (in other words, edges which belong to many short tours are the edges which receive the greater amount of pheromone). After the global updating, current set of ants 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 reach to the maximum number of turns (MAX_TURNS). Note that, we set the parameter N^{turns} such that, most of the ants in the initial set were able to reach the target point.

The pseudo-code of the ant colony algorithm is presented as follows:

```
1  Initialize
2   turn = 0, turnsRemaining = N^turns + 1
3  Loop
4   Release a new set of ants from the starting
    point
5   Loop
6     turn = turn + 1
7     turnsRemaining = turnsRemaining -1
8     For each ant 'a' in the current set
9       If ant 'a' does not reach to target point
10        Move to the next grid point using random
          propositional rule
11      Else
12        Ant 'a' stops exploring
13  Until (turnsRemaining = 0)
14  Apply the global pheromone update rule using
    ants that reached to the target point
15  Update optimal path best so far
16  Remove the current set of ants from the
    civilization
    turnsRemaining = N^turns + 1
17 Until (turn <= MAX_TURNS)
```

The state transition rule used by ant system, called a random-proportional rule (probabilistic state transition rule), is given by (1) and gives the probability with which ant k in city r chooses to move to the 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, $\eta = 1/\delta$ is the inverse of the distance (δ) from the 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 ($\beta > 0$).

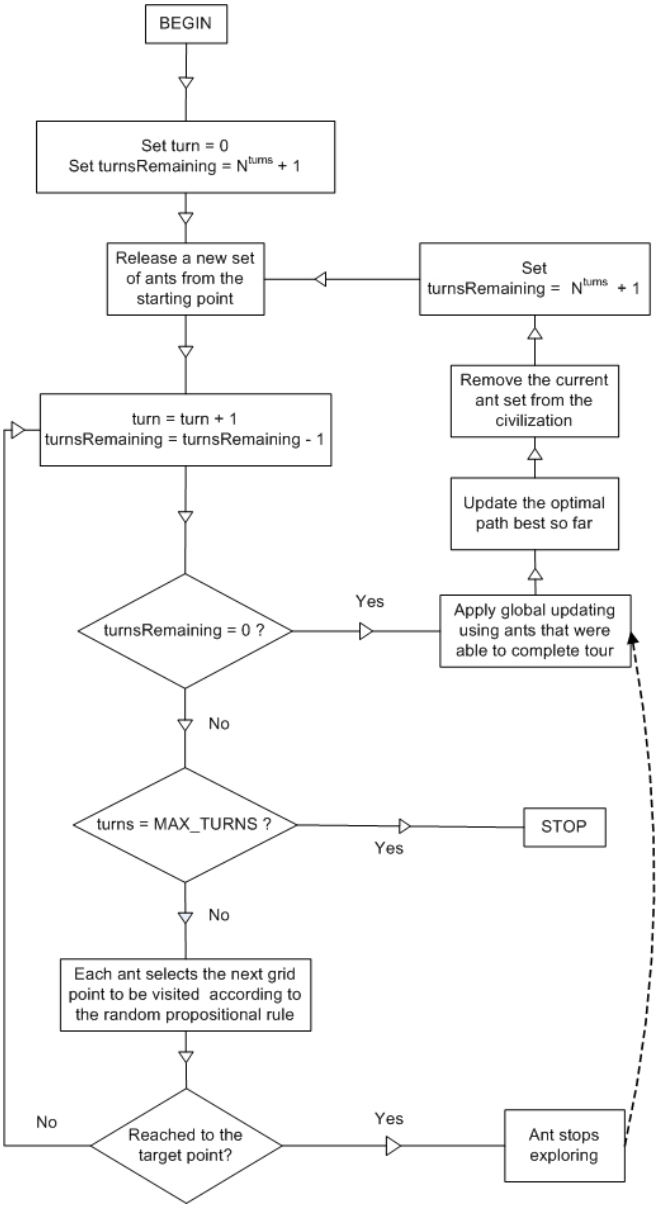


Fig. 2 Flow chart of the ant colony algorithm

In ant system, the global updating rule is implemented as follows. Ants that were able to complete their tour within the number of allocated turns (N^{turns}), allow to update 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

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

$0 < \rho < 1$ 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 turns N^{turns}.

Simple example for grid-based ant colony algorithm

Following simple example explains the grid-based ant colony algorithm in 2D environment.

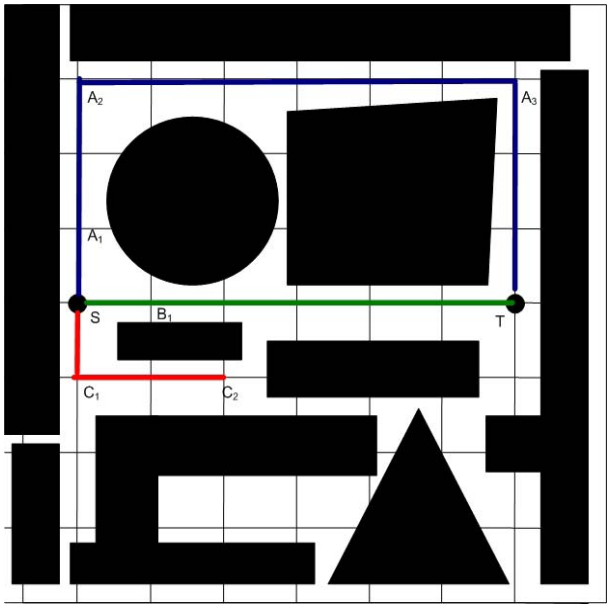


Fig. 3 Problem-solving of ants

Suppose that there 3 routes (two routes leading to the target point T and one route leading to the grid point C_2) from the start point S: R_1 ($SA_1A_2A_3T$), R_2 (SB_1T) and R_3 (SC_1C_2) such that length of R_1 is double as R_2. Assume that initially 10 ants are at the start point S and initial pheromone level for each edge is 100, $N^{turns} = 20$, $\rho = 0.01$, and $\beta = 5$. These ants have to select one of the paths R_1, R_2, and R_3 according to the random propositional rule (1). If the distance between two neighboring grid points is 1 unit, then $C_1T = A_1T = 6.08$ and $B_1T = 5$.

Probability of selecting the grid point A_1 (or route R_1) p_1 (from (1))

$$p_1 = \frac{100 \times \left(1/6.08\right)^5}{100 \times \left(1/6.08\right)^5 + 100 \times \left(1/6.08\right)^5 + 100 \times \left(1/5\right)^5}$$

$$p_1 \approx 1/5$$

Similarly, the probabilities of selecting the grid points B_1 (route R_2) and C_1 (route R_3): $p_2 \approx 3/5$ and $p_3 \approx 1/5$ can be calculated.

1. According to these probabilities, there are more chances to select route R_2 by 6 ants, R_1 and R_3 by 2 ants each.
2. After 20 turns ($N^{turns} = 20$), algorithm updates pheromone levels (2) as follows:
   a. Each edge on the route R_2
      $= (1 - 0.01) \times 100 + 6 \times (1/6)$
      $= 100$
   b. Each edge on the route R_1
      $= (1 - 0.01) \times 100 + 2 \times (1/12)$
      $= 99.17$
   c. Each edge on the route R_3
      $= (1 - 0.01) \times 100$
      $= 99$

   Since ants that followed the route R_3 get lost (unable to find the target point), the algorithm only evaporates the pheromone levels of the edges on that route.

In this example, the algorithm set a higher pheromone level for the shortest path (R_2) than the path (R_1). Further, the pheromone levels of lost path also decrease.

III. The Tessellated Representation and RAPID

The proposed algorithm accepts only the tessellated format of the solid model. Generating the tessellated format of solid model is supported by most of the CAD packages (including Pro/Engineer and AutoCAD). This helps the algorithm to accept any solid model generated by any CAD packages. Further, most of the collision detection programs (including RAPID) accept only the approximated triangular facets of the original model.

The .stl (STereoLithography) format or tessellated format [7] is an ASCII or binary file used in manufacturing. It is a list of triangular planes that approximates 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 the Fig. 4. This file is basically contained X, Y and Z cartesian coordinates of the each vertex of the triangle, as well as the coordinates of normal vector to the triangle. 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.

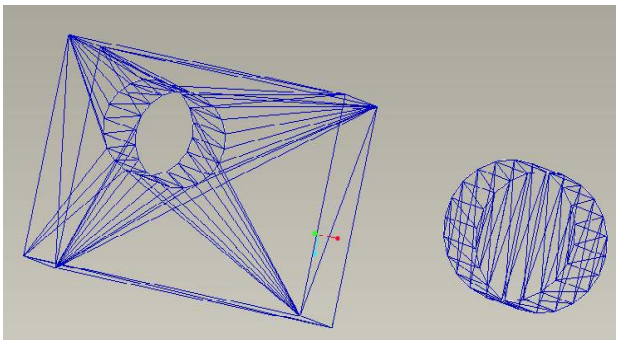


Fig. 4 Tessellated representation of objects

The proposed algorithm is based on 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 which 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, for some collision detection systems require the shapes to be well-formed solids – the surfaces must "closed" so that there is a well-defined inside and outside.

IV. Algorithm Implementation

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

- grid size
  Grid: [X^{min} (Δ_x) X^{max}, Y^{min} (Δ_y) Y^{max}, Z^{min} (Δ_z) Z^{max}]
  where (X^{min}, Y^{min}, Z^{min}) and (X^{max}, Y^{max}, Z^{max}) represents the minimum and maximum coordinates of the world where the paths should be explored and Δ_x, Δ_y, Δ_z defines the increment on x, y, z coordinates respectively,
- 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 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 from the previous work of 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 of time 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. Further, 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, program implements three tasks.

Firstly, it creates the whole road map using the rectangular grids. 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 centered on the line segment between the two points such that the cylindrical hose could be laid within it. 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 may reduce the number of routes needs to be stored in the memory. The road map was stored in a text file which can be used again if the algorithm needs to be executed another time.

Secondly, the program searched for the optimal path or near optimal path between the start and the target points using the ant colony and the road maps created earlier. If path contained cycles, these were removed before applying global updating of the pheromone. Initially, a constant pheromone value was set for each edge. Before applying the 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 sets this path as the optimum path found so far.

Thirdly, at the end of MAX_TURNS, the (optimal) path obtained was further refined to eliminate some 'staircases'

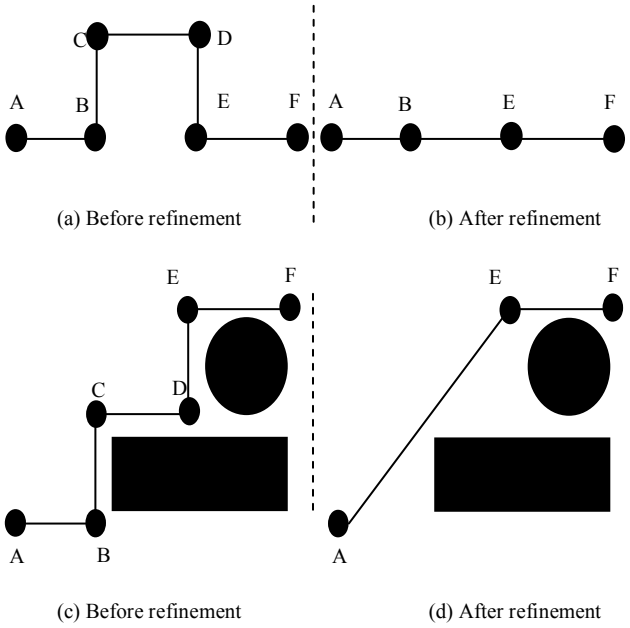
(See Fig. 5). Again, when refining the optimal path, before connecting two points, the algorithm RAPID was used to detect any collisions.

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

In the final step, the list of points needs to be connected for the optimal path passes into CAD software for drawing the pipe segments.

V. Simulations

The effectiveness of the algorithm was demonstrated by simulations. CAD package Pro/Engineer was used for generating the 3D models and its programming toolkit Pro/Toolkit was used for obtaining the tessellated format of the generated models.



(a) Before refinement (b) After refinement

(c) Before refinement (d) After refinement

Fig. 5 Refining the path

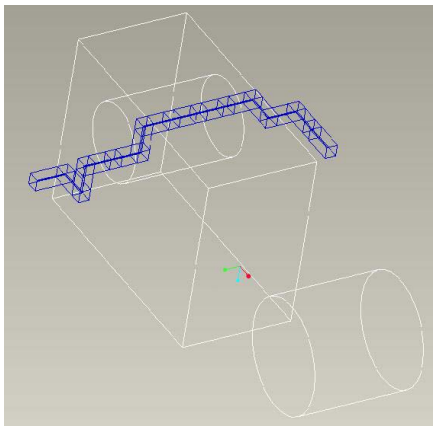
The parameter settings for the ant colony algorithm were: the number of ants = 10, initial pheromone level for each edge = 100, number of turns for which the algorithm is to be run, MAX_TURNS = 10,000, pheromone decay parameter ρ = 0.01, and β = 5. Sizes of the grids of the world are selected depending on the problem at hand.

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).

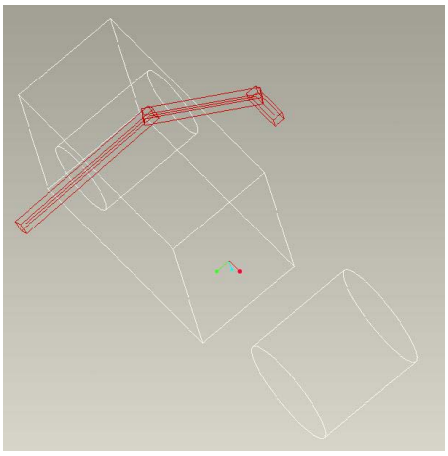
The following simulations were carried out for testing the efficiency of the propose algorithm.

A. Hose routing in an environment with a hole in a cube

The proposed grid base ant colony algorithm was tested in an environment consisting of cube containing a hole (Fig. 6). Hose segments needed to be laid inside this hole in order to obtain the optimal path. Fig. 6(a) shows the optimal path generated by ant colony algorithm. As the algorithm was used only north, east, west, south, top, and bottom neighbour points, this optimal path had some 'staircases' like pipe segments. Fig. 6(b) shows the refined path which is the optimum path and number of bends reduced to 2 from 8.



(a) Optimum path given by the ant colony algorithm



(b) Refined path

Fig. 6 Hole in a cube
{Grid: [-250 (25) 150, -50 (25) 150, -200 (25) 0];
S: (-200, 150, -100); T: (-100, -50, -150);
Radius: 5; N^{turns}: 100; Time: 29 secs.}

Fig. 7 shows the obtained optimal path for the same 3D model as in Fig. 6, but the Δ values are doubled. However, the time taken to find the path was reduced; it is not the desired optimal path. This simulation shows that selection of the right grid size is an important part of the algorithm.

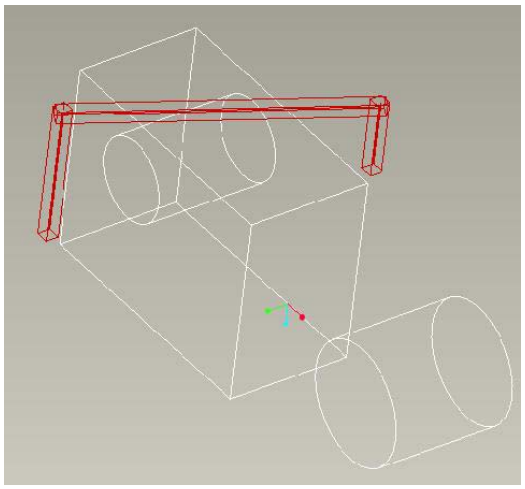


Fig. 7 Hole in a cube with a larger grid size
{Grid: [-250 (50) 150, -50 (50) 150, -200 (50) 0];
S: (-200, 150, -100); T: (-100, -50, -150);
Radius: 5; N^{turns}: 100; Time: 2 secs.}

B. *Hose routing in an environment with a hole in a cube and the optimal path is block with 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. 8).

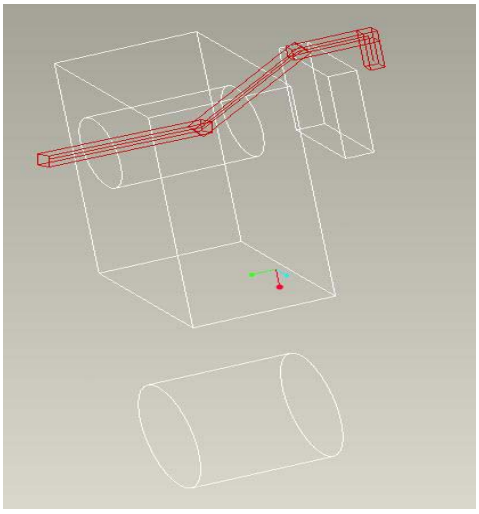


Fig. 8 Optimal path is blocked by a cubic obstacle
{Grid: [-250 (25) 150, -50 (25) 150, -200 (25) 0];
S: (-200, 150, -100); T: (-200, -100, -150);
Radius: 5; N^{turns}: 100; Time: 39 secs.}

C. *Hose routing in an environment with a U-shape obstacle*

In this experiment, a U-shape obstacle was placed in the environment and the environment was made more complex by introducing other objects (see Fig. 9). Furthermore, the start and target points were placed such that only one path existed between them. Note that, z coordinates of the search space were restricted to the top and bottom of the obstacles.

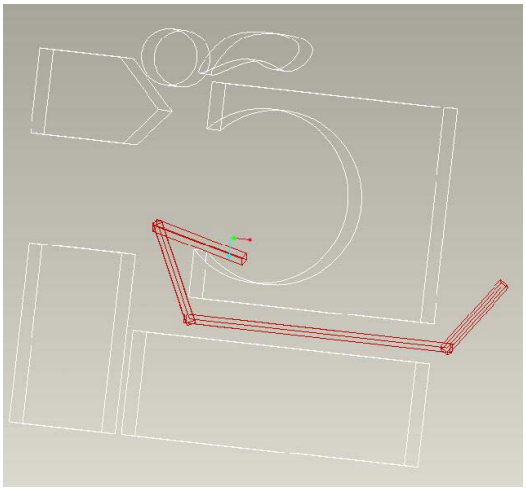


Fig. 9 U-shaped obstacle
{Grid: [-300 (25) 400, 0 (25) 100, -300 (25) 400];
S: (50, 25, -50); T: (350, 25, -50);
Radius: 5; N^{turns}: 100; Time: 109 secs.}

D. *Hose routing in an environment with a U-shape obstacle and the optimal path is block with two obstacles*

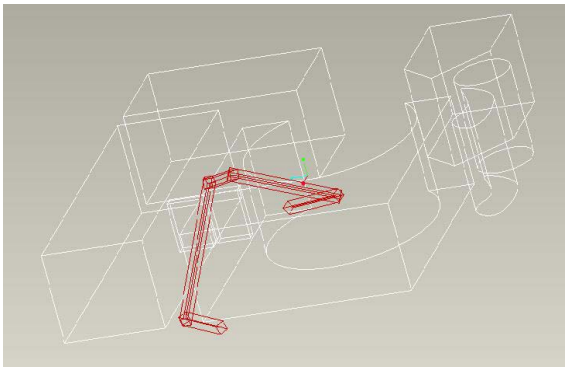


Fig. 10 U-shaped obstacle and optimal path is blocked by two obstacles
{Grid: [-300 (25) 400, 0 (25) 100, -300 (25) 400];
S: (50, 25, -50); T: (350, 25, -50);
Radius: 5; N^{turns}: 100; Time: 117 secs.}

In this simulation, the optimal path found in the earlier case was blocked by two cubic obstacles (see Fig. 10).

E. *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. 11).

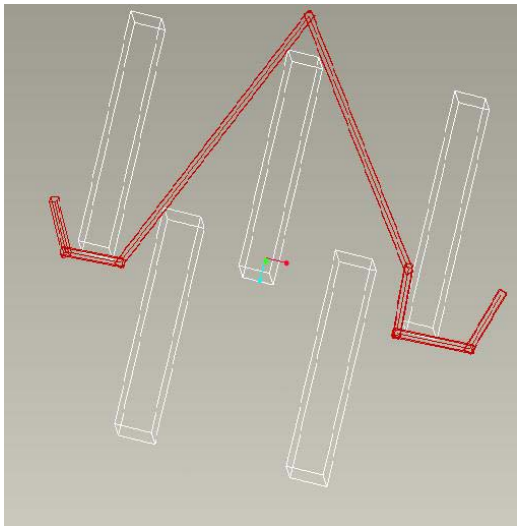


Fig. 11 Parallel walls
{Grid: [-300 (25) 300, 0 (25) 100, -300 (25) 300];
S: (-300, 25, 0); T: (300, 50, -25);
Radius: 5; N^{turns}: 200; Time: 162 secs.;
MAX_TURNS = 20,000; No of ants = 20}

VI. Discussion

Previously, scientists have applied the ant colony algorithm for many real-world problems such as travelling salesman problem, 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 rectangular grid.

The problem presented in this paper and the TSP are quite similar; however there are also 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 to a point, it must not visit that point again. To this, 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 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 experiment described above, this is not always possible, as some 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 strength of the propose grid based ant colony algorithm for automatic 3D hose/pipe routing. Simulation results show that the algorithm can be applied for any shape which can be generated using any CAD package.. The use of the RAPID library greatly helps the algorithm to detect collisions when laying the hoses.

Simulation study also indicates that the proposed grid based ant colony algorithm is of practical use because the required computational times are reasonably low.

However, 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, algorithm fails to obtain the optimal path (See Fig. 6 and Fig. 7). Thus, selecting the right size of the grid is an important part of the algorithm.

VII. Conclusion

In this paper, a grid-based ant colony algorithm has been proposed for automatic 3D hose routing. The algorithm generates the optimal set of the 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 handle only the triangular shapes. However, the accuracy of the collision detection depends on the number of triangles used to approximate the obstacles. The effectiveness 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 package. The computational efficiency suggests that the algorithm can be applied to real-world hose/pipe routing problems.

The selection of the right resolution (or size of the grid) plays an important part of the algorithm and it is dependent on the problem at hand (See Fig. 6 and Fig. 7). When the resolution is increased, the algorithm requires higher amount of memory and more time to compute the results. The algorithm can be improved, if the domain knowledge of the problem is incorporated into the algorithm. For this, selecting random points from the free space and use them to create the road map will be investigated. Another solution to this problem is use of multi-resolution algorithm instead of uniform resolution.

At the initial stage of the experiment, the algorithm has been implemented only for optimizing the distance between the start and the 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 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 thank 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). <http://www.cs.unc.edu/~geom/OBB/OBBT.html>.