Defence Science Journal, Vol. 65, No. 3, May 2015, pp. 220-225, DOI : 10.14429/dsj.65.7855

Simulation for Path Planning of SLOCUM Glider in Near-bottom Ocean Currents Using Heuristic Algorithms and Q-Learning

Utkarsh Gautam* and Malmathanraj Ramanathan!

*Department of Instrumentation and Control Engineering, JSS Academy of Technical Education, Noida
!Department of ECE, National Institute of Technology, Trichy
*E-mail:ugautam_91@yahoo.com

ABSTRACT

Addressing the need for exploration of benthic zones utilising autonomous underwater vehicles, this paper presents a simulation for an optimised path planning from the source node to the destination node of the autonomous underwater vehicle SLOCUM Glider in near-bottom ocean environment. Near-bottom ocean current data from the Bedford Institute of Oceanography, Canada, have been used for this simulation. A cost function is formulated to describe the dynamics of the autonomous underwater vehicle in near-bottom ocean currents. This cost function is then optimised using various biologically-inspired algorithms such as genetic algorithm, Ant Colony optimisation algorithm and particle swarm optimisation algorithm. The simulation of path planning is also performed using Q-learning technique and the results are compared with the biologically-inspired algorithms. The results clearly show that the Q-learning algorithm is better in computational complexity than the biologically-inspired algorithms. The ease of simulating the environment is also more in the case of Q-learning techniques. Hence this paper presents an effective path planning technique, which has been tested for the SLOCUM glider and it may be extended for use in any standard autonomous underwater vehicle.

Keywords: Simulation, path planning, AUV SLOCUM Glider, near-bottom ocean currents, Q-learning, genetic algorithm, ant colony optimisation algorithm, particle swarm optimisation algorithm

1. INTRODUCTION

With the recent development in technology, autonomous underwater vehicles (AUVs) are increasingly being used to efficiently explore the various natural and artificial environments present on seabed. The AUVs are efficient and effective for a variety of missions within the depths of waterbodies. One such AUV is the AUV SLOCUM Glider (ASG), manufactured by the Teledyne Webb Research[2]. This AUV boasts the following features: dynamic buoyancy, seven hundred and twenty hours of endurance at nominal load, and a capacity to work at one thousand meters of depth[2]. Addressing the need of the study of benthic zones, this paper develops a simulation for the path planning of the ASG so that it may be used in near-bottom ocean currents. Defining the near-bottom ocean current are the Gully marine protected area (MPA) under Canada's Oceans Act, the Gully becomes Canada's second Oceans Act MPA, and the first in the Atlantic region.

The Gully MPA regulations and accompanying regulatory impact analysis statement can be viewed[13]. The data for the magnitude and direction of the velocity of the ocean currents at near-bottom ocean depth is taken from the Bedford Institute of Oceanography[3]. The simulation is done using MATLAB platform. Using the above data, a cost function was developed which accurately describes the dynamics of the ASG and its interaction with the near-bottom ocean environment. Cost function is optimised to obtain the shortest path between the source node and destination node. This optimisation was achieved using various biologically-inspired algorithms such as genetic algorithm (GA), ant colony optimisation (ACO) algorithm and particle swarm optimisation (PSO) algorithm. Optimisation was also done using Q-learning technique and the results obtained from both the biologically-inspired algorithms and Q-learning techniques were compared. The results clearly show that the Q-learning technique is computationally less expensive as compared to biologically-inspired algorithms, also using Q-learning the environment can be simulated much easily.

Since the advent of AUVs much work has been done to improve their path planning and obstacle-avoidance abilities. Eichhorn[6-9], *et al.* published a series of works pertaining to the path planning and obstacle-avoidance of the ASG. They mainly used graph-based methods for the same. The gliders used in International scenario are gulper, wave, spray, and sea glider. Aghababa[4] utilised various evolutionary algorithms for path planning and collision avoidance of an underwater vehicle[9]. The nature of computing of ACO algorithm was introduced by Dorigo[5], *et al*. Shi and Eberhart[12] gives much insight on the PSO algorithm. The research works is a result of an inspiration from the above-mentioned works and has proposed a method for path planning of the ASG using tabular Q-learning technique and have compared it with other optimisation algorithms.

Received 18 July 2014, revised 11 May 2015, online published 29 May 2015

2. COST FUNCTION

A cost function[10] was required to accurately describe the dynamics of the ASG and its interaction with the near-bottom ocean environment, especially with the near-bottom ocean currents. The work is inspired from the works of Eichhorn[6-9] and this research paper develops a cost function which takes into consideration the dynamics of the ASG, its principle of locomotion, and the spatially-varying velocity of ocean currents at near-bottom depths of the ocean. The cost function takes the input as the coordinates of the source node and the coordinates of the destination node and gives the output as time taken to travel from the source node to destination node.

The time taken from source node to destination node denoted by T is found using the equation of kinematics:

$$S=UT+\frac{1}{2}aT^2 \tag{1}$$

where S is the displacement, U is the initial velocity, a is the acceleration, and T is the time. Now as the ASG[2] has low cruising speed (0.2 ms^{-1} - 0.4 ms^{-1}) , taking acceleration, a to be zero, this gives as

$$T=\frac{S}{U} \tag{2}$$

here T is the time taken from source node to destination node, S is the distance between the source node and destination node, and U is the average velocity with which the AUV travels from the source node to destination node.

2.1 Ocean Current Determination

The data for the near-bottom ocean current velocity is obtained from Bedford Institute of Oceanography. The data consist of the magnitude and direction of the ocean current velocity at various latitudes, longitudes and depths near the bottom surface of the ocean. These data are processed to obtain the values suitable for creating a realistic simulation. Following processes are involved in obtaining of the ocean current values:

- Obtaining the data regarding the near-bottom ocean currents from Bedford Institute of Oceanography[3]
- Segregating the data to obtain separate databases of ocean current velocity values for different latitudes, longitudes, and depths respectively.
- Identifying the region of interest.
- Interpolating the separate databases of latitude, longitude and depth, varying ocean current values using one-dimensional interpolation functions within the range of the region of interest.
- The ocean current value at any particular coordinate is obtained by averaging the interpolated value of the ocean current at the required latitude, longitude, and depth.

As the data obtained has ocean current values at haphazard coordinates, it is necessary to interpolate within the range of region of interest to obtain ocean current values at regular intervals of the coordinates, thereby making the simulation more accurate.

2.2 Determination of Average Velocity

The velocity of the ASG at a particular coordinate, denoted by V_{ef} depends on its cruising speed, denoted by V_c, the magnitude and direction of the ocean current velocity, denoted by $V_{current}$ and the direction of the path from the previous node to the present node, denoted by V_{path} (for the source node the previous node is taken as the origin). Considering the shape of the ASG, the interaction of the ocean currents with it can be approximately modelled as an intersection between a line and a circle or a sphere[4]. The point of intersection gives the V_{ef} as clarified by the following Eqn:

$$\text{Line: } x(V_{ef}) = V_{ef} V_{path} \tag{3}$$

$$\text{Circle/Sphere: } \quad V_c^2 = ||x - V_{current}||^2 \tag{4}$$

$$D = (V_{path}^T . V_{current})^2 + V_c^2 - V_{current}^T . V_{current} \tag{5}$$

The discriminant given by Eqn. (5) determines whether a particular path should be completely avoided or not, if the D becomes negative V_{ef} has no real value, hence, it would be Not *a* Number, *NaN*.

$$V_{ef} = V_{path}^T + \sqrt{D}, \text{ for } D > 0 \tag{6}$$

$$V_{ef} = NaN, \quad \text{otherwise.} \tag{7}$$

The average velocity $U(t)$ is determined by taking the average of the V_{ef1} and V_{ef2}, which are the effective velocities at source node and destination node, respectively.

$$U(t) = \frac{(V_{ef1} + V_{ef2})}{2} \tag{8}$$

2.3 Calculation of Distance

The distance between the source node and the destination node is calculated using the distance formula.

$$S = \sqrt{((x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2)} \tag{9}$$

where *x1, y1, z1* are the coordinates of source node and *x2, y2, z2* are the coordinates of destination node.

2.4 Dive Profile

The ASG changes its buoyancy to either ascend or descend within the waterbody. This defines its locomotion principle. Owing to its locomotion principle, the ASG has a characteristic dive profile which is sinusoidal in shape when observed in coarser approximation. As simulating the exact dive profile is computationally complex, we have approximated it and the result is a saw tooth shaped profile. For the sake of simulation we sample this dive profile and the discrete coordinates, thus obtained are given as input to the optimisation algorithms to establish shortest paths traversing all these points. By doing so we make sure that the characteristic dive profile of the ASG is accommodated in the path planning. The results presented in the paper also show that the optimised paths obtained from the algorithms are in sync with the dive profile of the ASG.

Before the cost function is optimised using the various heuristic algorithms, it is first optimised within itself. The ASG can divide its path from a source node to a destination node into various saw tooth shaped segments owing to its dive profile, a function within the cost function calculates the various time taken for the number of segments from one to the number of segments which is maximum. The minimum of the time taken is chosen as the final output of the cost function. As the

division of a single path into *n* number of segments can be both advantageous and disadvantageous for the ASG depending on whether it avoids highly turbulent ocean currents or not, the pre-optimisation takes care that all the possible cases are considered, hence, making the simulation more realistic and the path planning more efficient.

3. OPTIMISATION ALGORITHMS

Once the cost function is developed, various optimisation algorithms are used to minimise the cost function that is to minimise the time taken from source node to destination node and hence plan the shortest possible path from the source to the goal. Sampled discrete coordinates from the dive profile are chosen randomly and given to the optimisation algorithms. The results are observed to obey the dive profile of the ASG. These results are then compared. The inference from these comparisons is elucidated in the paper.

3.1 Genetic Algorithm

Genetic algorithm uses the techniques such as inheritance, mutation, selection and crossover recurrently till only the fittest individual, which is analogous to the best solution to the cost function remains and other less fit individuals or less optimised solutions wither away. For implementing the genetic algorithm firstly, a cost function must be defined that can evaluate the fitness of a solution. The potential solutions must be represented in a particular format (binary, real, gray coded, etc.). The Genetic Algorithm considers these potential solutions of the cost function as the genes of a chromosome and keeps modifying these randomly to get the fittest gene that is the optimised solution to the cost function[11]. The pool of potential solutions is called a population. This population generally consists of randomly generated solutions for the cost function. The crossover operator selects 'parents' from this population using Roulette Wheel method[11] and randomly recombines these to produce 'offspring' which are also valid solutions of the cost function. If these 'offspring' solutions yield better results, these survive else these are discarded. The mutation operator also randomly selects a candidate from the population and makes random modifications to it, again this mutated solution is checked using the cost function, and if it yields optimised results, it survives, else it is discarded. Thus, imitating the principles of natural evolution, the most optimum result is selected by the genetic algorithm.

3.2 Ant Colony Optimisation Algorithm

Ant colony optimisation has been inspired by the inherent ability of the ants to find the shortest distance between their nest and their goal, which is mostly the food. Ants leave a trail of pheromone deposits wherever they travel. Assume a case where one ant leaves the nest in search of food, it takes random steps and finally reaches the food. Now, all along those random paths pheromone has been deposited. Now, as is obvious, once the ant has reached the goal, while returning, it takes the shortest path, therefore the amount of pheromone deposited in the shortest path is more as compared to other random paths. Now when the subsequent ants travel to the goal, they sense the path with the maximum amount of pheromone. By doing so not only do they find the shortest path but also deposit more pheromones on the shortest path compensating for any loss in pheromone deposits caused due to evaporation. To utilise this trait of an ant colony in computationally optimising cost functions, first a set of *m* random solutions to the cost function is chosen, these *m* random solutions represent *m* ants. Now each ant (solution) is evaluated on the basis of the output it begets from the cost function. According to this evaluation pheromone deposit concentration linked with the route taken by each ant is modified using the following equation[5]:

$$\tau_{ij}(t) = \rho\, \tau_{ij}(t\text{-}1) + \Delta\, \tau_{ij};\ t = 1,2,3\ldots T \tag{10}$$

where T is the number of iterations, $\tau_{ij}(t)$ is the revised concentration of pheromone associated with path option I_{ij} at iteration t; $\tau_{ij}(t\text{-}1)$ is the concentration of pheromone at previous iteration $(t\text{-}1)$, $\Delta\, \tau_{ij}$ is the change in pheromone concentration and ρ is the pheromone evaporation rate having a value between zero and one. $\Delta\, \tau_{ij}$ is calculated using the following equation:

$$\Delta\tau_{ij} = \sum_{k}^{m} = 0 \frac{R}{fitness k} \text{ if option } I_{ij} \text{ is chosen by ant } k$$
$$\Delta\, \tau_{ij} = 0 \qquad \text{otherwise.} \tag{11}$$

Here R is a constant called the pheromone reward factor and *fitnessk* is the value of inverse of the cost function for the k^{th} ant (solution). After the pheromone deposit concentration is refreshed for one iteration, a different path is chosen for the next iteration, this choice of path is randomised using a roulette method of selecting the path.

3.3 Particle Swarm Optimisation Algorithm

Particle swarm optimisation algorithm owes its origin to the behaviour of the birds in a flock while flying to reach a particular destination. While flying in the search space, each bird of the flock looks in a particular direction and also each bird keeps on communicating with the other birds of the flock. This communication helps the birds to identify the one member of the flock which is at the best location with respect to the goal. Once this bird is identified, other birds fly towards the best location with a velocity that depends on their current position. After reaching the new positions, the birds again infer their positions to figure out the best position, this process is repeated until the flock reaches its destination. In this paper, authors have utilised the global optimising model proposed by Shi and Eberhart[12], given as:

$$V_{i+1} = w * V_i + rand * C1 * (P_{best} - x_i) + rand * C2 * (G_{best} - x_i) \tag{12}$$

$$x_{i+1} = x_i + V_{i+1} \tag{13}$$

here V_i is the velocity of the i^{th} member of the flock, x_i is its position, *C1* and *C2* are positive constant parameters called the acceleration coefficients, rand is uniformly distributed random number generating functions that generate random numbers in the range [0 1], P_{best} is the best position of the i^{th} particle and G_{best} is best position among all particles of the flock and w is the inertial weight. Weight factor used is given by,

$$W_{i+1} = w_{\max} - \frac{w_{\max} - w_{\min}}{N_{iteration}} * i \tag{14}$$

where $N_{iteration}$ is the maximum number of iterations and $w_{\max}$ and $w_{\min}$ are maximum and minimum values of w, respectively.

3.4 Q-Learning Technique

Q-learning is a form of reinforcement learning which further belongs to the super class of machine learning. A computer system is said to learn from a data set denoted by D to perform the task denoted by T if after learning the system's performance on T improves as measured by a performance measurement index, denoted by M. Reinforcement learning utilises trial and error method to learn, it takes random actions to make a system achieve its goal and then it obtains feedback in the form of rewards or penalties that determine whether the action taken was correct or incorrect, therefore, after a number of iterations, the algorithm learns which steps are beneficial and which are not wrt to the goal to be achieved. This paper utilises Tabular Q-learning Technique to optimise the path between the source node and goal node. To implement the Q-learning, first the cost function is converted to a matrix form, termed as the reward matrix and denoted by R. The Q matrix was initialized as all zeroes. Now for a large number of iterations, the Q matrix is modified using the following equation:

$$Q(\text{state, action}) = R(\text{state, action}) + \gamma . \text{Max}(Q(\text{next state, all actions})) \quad (15)$$

The initial state is the coordinates of the source node and the next state is chosen randomly. Then for all possible paths, from the next state to the goal state, are checked and given rewards accordingly. After some iterations, the coordinates of the optimised path have the highest Q values, and hence, are chosen as the optimised path.

4. RESULTS AND DISCUSSIONS

Random points sampled from the dive profile are given to the GA. These points include the starting node and goal node. The GA is programmed in MATLAB in such a way that it finds the shortest distance with reference to the cost function of the ASG, from the starting node to goal node, whilst traversing all the given random points and also finding its way back to the starting node. The results obtained show that not only does the GA find the shortest path, the path thus obtained also replicates the dive profile of the ASG. The time taken by the GA to compute the optimal function was 12.765700 CPU seconds. Figure 1 clearly shows the optimised path generated by the GA. Similarly, random points sampled from the dive profile are given as input to the ACO algorithm. The starting node in this case is taken as origin. The ACO also gives a similar result with shortest path being evaluated incorporating all the random points given as input. The resulting path also was as expected from the dive profile of the ASG (evident from Fig. 2).

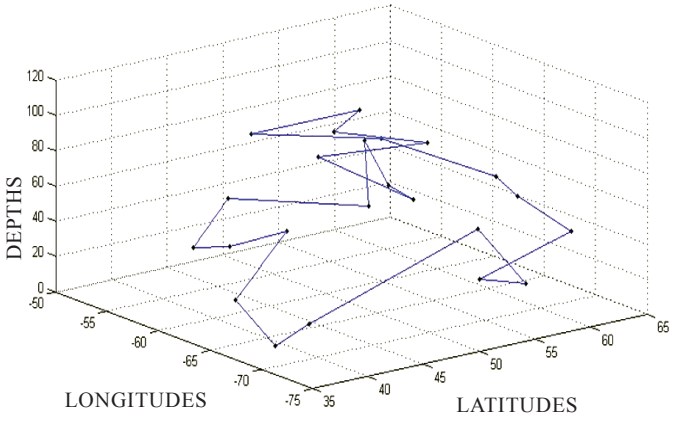


Figure 1. Glider underwater flight profile optimised using genetic algorithm.

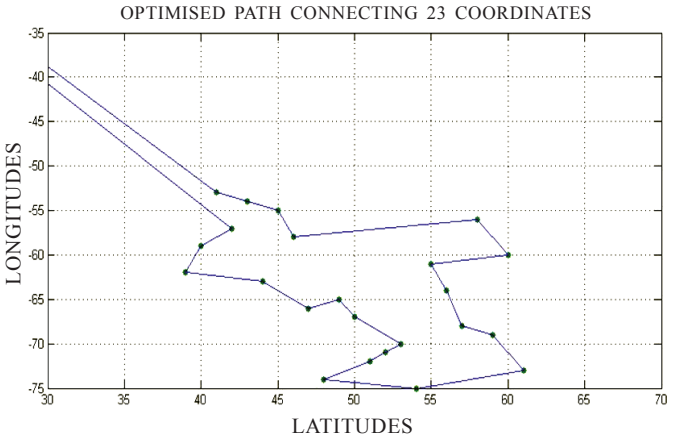
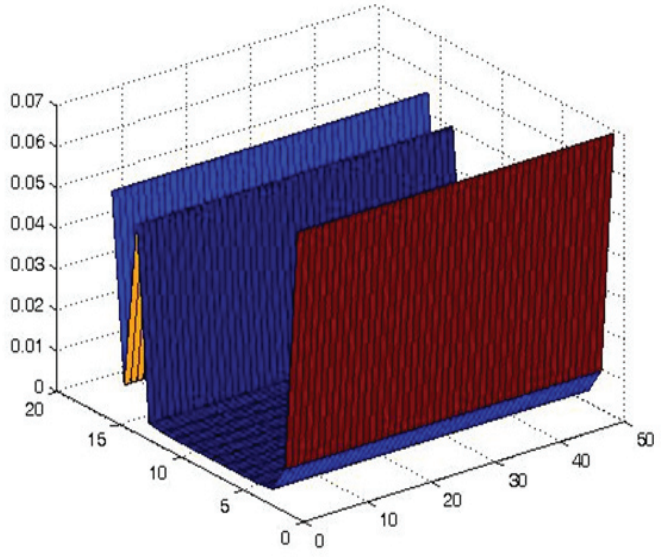


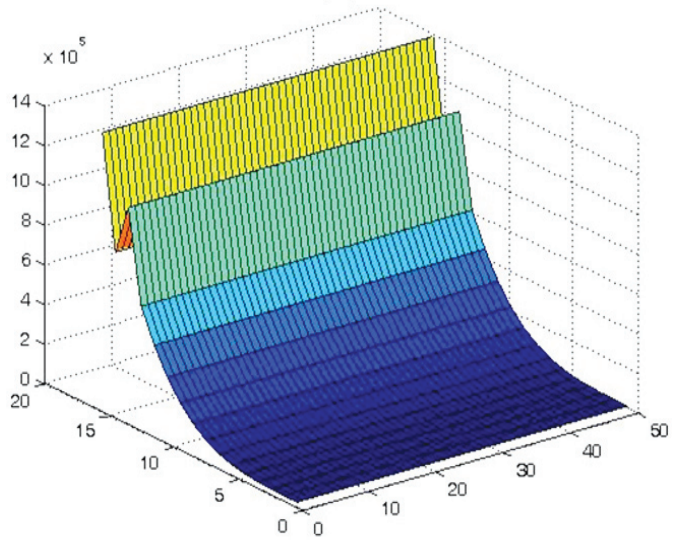
Figure 2. Glider underwater flight profile optimised using ant colony optimisation.

The time taken by the ACO to compute the optimal function was 120.280000 CPU seconds. The Fig. 2 showcases the optimised path generated by the ACO algorithm. The value of ρ is taken as 0.15. The numbers of ants used in the simulation was 25 and the value of R used was 0.33. Figures 3, 4, and 5 show the impact of the PSO algorithm in optimising Foxhole, Dejong F4 and f6 functions, respectively. It is evident from Fig. 6 that the cost function reaches saturation as the generation number increases. The number of particles of the swarm is taken as 40, the values of $C1$ and $C2$ are set as 2.5 each, the value of w_{max} is 0.1 and that of w_{min} is 0.9. Figure 7 shows the maximum Q value of path obtained after standard one thousand iterations in red colour and the other Q value of the tabular column shown in different colours. The coordinates



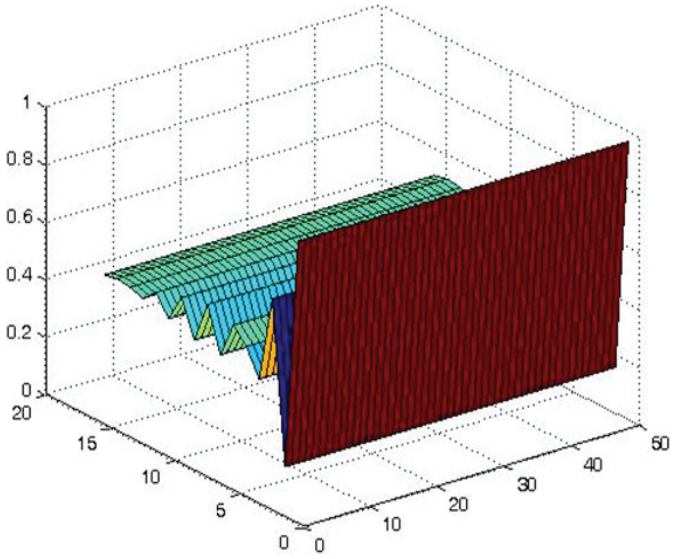
X axis - X, Y axis- Y, Z axis – F(X,Y)

Figure 3. Foxhole function optimised by PSO.



X axis - X, Y axis – Y, Z axis – F(X,Y)

Figure 4. Function Dejong F4 optimised by PSO.



X axis - X, Y axis – Y, Z axis – F(X,Y)

Figure 5. Function f6 optimised by PSO.

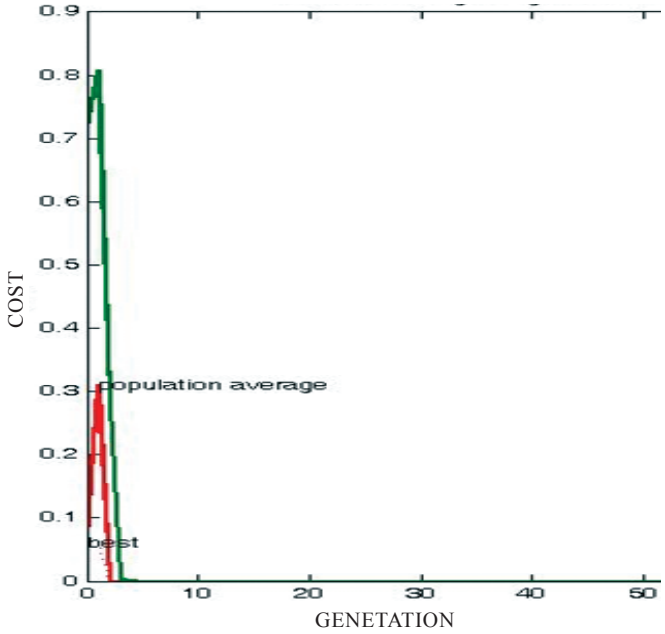


Figure 6. Plot of cost function *vs* number of generations.

corresponding to the maximum Q values (in the Q matrix) depict the shortest path with reference to the cost function as optimised by the Q-learning technique. The computational time taken by the Q-learning was 7.466598 CPU seconds. It is also seen from Fig. 7 that the Q values have saturated for only the optimised path after a standard number of iterations. The γ is used with a constant value of 0.8.

5. CONCLUSION

It can be concluded from the paper that the Q-learning approach is better in path planning of the ASG wrt to computational complexity and ease of environment simulation. The paper, therefore presents an effective method of path planning using tabular Q-learning technique. Concept of function approximation can be used instead of tabular Q-learning so as to improve computational complexity.

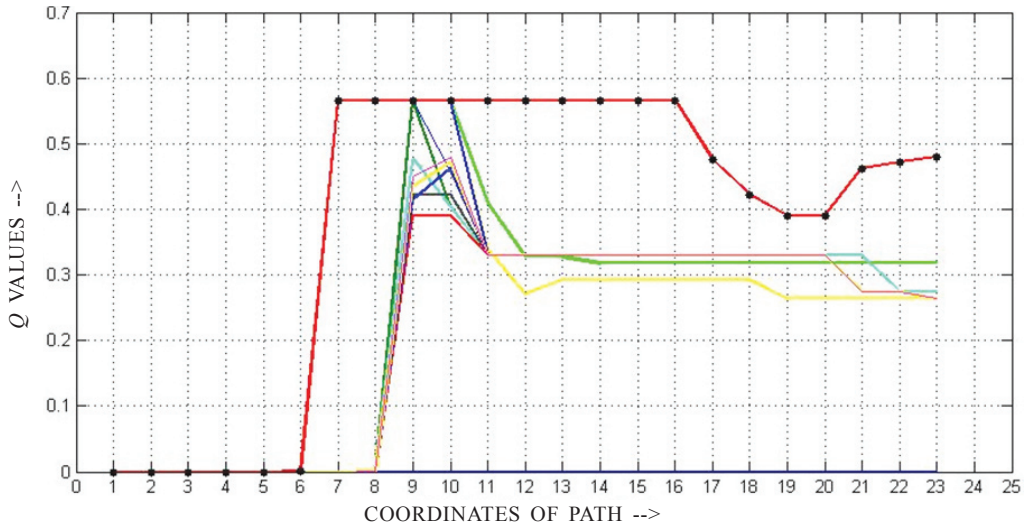


Figure 7. Plot of Q values till 1000 iterations *vs* coordinates of path.

REFERENCES

1. http://noc.ac.uk/research-at-sea/exploration-at-sea/ocean-floor [Accessed on 20th June 2014].
2. www.auvac.org [Accessed on 20th June 2014].
3. http://www.bio.gc.ca/science/data-donnees/archive/current_statistics/bottomcurrents-eng.php [Accessed on 20th June 2014].
4. Aghababa, M.P. 3-D path planning for underwater vehicles using five evolutionary optimisation algorithms avoiding static and energetic obstacles. *Appl. Ocean Res.,* 2012, **38,** 48-62. doi: 10.1016/j.apor.2012.06.002
5. Dorigo, M.; Maniezzo, V. & Colorni, A. Ant system: optimisation by a colony of cooperating agents. *IEEE Trans. Sys., Man Cyber.*, *Part B: Cybernetics*, 1996, **29,** 26-41. doi: 10.1109/3477.484436
6. Eichhorn, M. A new concept for an obstacle-avoidance system for the AUV SLOCUM Glider operation under ice. *In* Oceans'09 IEEE, Bremen, Germany, 2009. doi: 10.1109/oceanse.2009.5278350
7. Eichhorn, M. Optimal path planning for AUVs in time varying ocean flows. *In* 16th Symposium on Unmanned Untethered Submersible Technology, UUST09, Durham, NH, USA, 2009.
8. Eichhorn, M; Williams, C.D.; Bachmayer, R. & deYoung, B. A mission planning system for the AUV Slocum Glider for the Newfoundland and Labrador Shelf. *In* Oceans'10 IEEE, Sydney, Australia, 2010. doi: 10.1109/oceanssyd.2010.5603919
9. Eichhorn, M. Solutions for practice – oriented requirements for optimal path planning for AUV SLOCUM Glider. *In* Oceans'10, IEEE, Seattle, USA, 2010. doi: 10.1109/oceans.2010.5664082
10. Joshua, Grady Graver. Underwater gliders: Dynamics, control and design. Princeton University, 2005. (PhD, Dissertation)
11. Man, K.F.; Tang, K.S. & Kwong, S. Genetic algorithms: Concepts and applications. *IEEE Trans. Industrial Electro.,* 1996, **43**(5). doi: 10.1109/41.538609
12. Shi, Y. & Eberhart, A.R. Modified particle swarm optimiser. *In* Proceedings of the IEEE International Conference on Evolutionary Computation, 1998. 69-73. doi: 10.1109/icec.1998.699146
13. http://canadagazette.gc.ca/partII/2004/20040519/html/sor112-e.html [Accessed on 7th May 2015].

CONTRIBUTORS

Mr Utkarsh Gautam was involved in the Simulation of Autonomous Underwater Vehicle (AUV) SLOCUM Glider for an optimized path planning using heuristic algorithms.

Mr Malmathanraj Ramanathan was involved with implementing the Q-Learning algorithm, benthic zones and analysing the cost function modeling for the AUV.