International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 7, Issue 1, 2020, PP 29-36
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
DOI: http://dx.doi.org/10.20431/2349-4859.0701004
www.arcjournals.org

Reverse-Twister Swarm Search Algorithm Design: NASA Swarmathon Competition

Tariq H. Tashtoush*, Roger Hernandez, Raquel Yanez, Jorge Gonzalez Jr., Hector Moreno, Valeria Escobar

Texas A&M International University, Laredo, Texas, USA

Corresponding Author:*Tariq H. Tashtoush, *Texas A&M International University, Laredo, Texas, USA*

Abstract:*The development of search algorithms is essential to optimize space exploration. The use of swarms search robots can optimize exploration techniques. Ourteam (DustySWARM NASA Robotics) developed a reverse-twister code search algorithm that can be used to control a swarm of autonomous robots. The final version of the Twister code indicated an increase in the volume of resources collected by the swarm of robots within the predefined time.*

Keywords:*Swarm Robotics, Searching Algorithm, Autonomous, Robot Swarm. Robot Operating System (ROS), NASA Space Exploration, Mars Mining, Simulation, Autonomous Robot Swarm; Collaborative robots.*

Public Interest Statement: *Robots are the future that will assist human beings to concur with their dreams of exploring the world and space beyond the Moon. With the current limitations that hinder the ability of astronauts to travel long distances to explore Mars, NASA started an effort to develop robotics systems to revolutionize space exploration. This competition aims to develop a cooperative roboticssystem that includeshardware and software to support the work done by NASA's Human Exploration and Operations Mission. Engineering students came together to be creative and solve such a complex problem and help to further advance technology for future NASA space exploration missions.*

1. Introduction

To revolutionize space exploration techniques, the DustySWARM team from Texas A&M International University took the challenge to improve current search algorithms. The focus of the research was to create an efficient search algorithm applicable to the future mission of NASA space exploration. The use of autonomous search rovers that can react to their environment facilitates the exploration of unknown territories. Furthermore, swarms of autonomous robots reduce the data and resource collection period. In nature, swarms of animals and insects have developed, by instinct, searching systems to collect resources for food and shelter. Within the groups, the ability to achieve their goal of resource collection is obtained by systems of communication and reactions to the environment. Similarly, robot swarms can collaborate to explore unknown terrain. Previous search algorithms have been developed, such as the iAnt code developed by the University of New Mexico, which implements the use of sensor feedback and programmed decision logic to search for and collect resources.

The Reverse-Twister code, presented in this paper, was developed by the DustySWARM team to compete in the first NASA Swarmathon Competition 2016. The Reverse-Twister code created for the competition implements an approximation of an Archimedean Spiral. The purpose of the competition is to expand student knowledge in robotics and simultaneously improve technology in space exploration missions. The results of the project indicated the possibility of more efficient space explorations in the future.

The paper is organized as follows: Section 2 is a background and literature review, Section 3 describes the model and algorithm development, Section 4 deals with the analysis and development of the experimental simulation runs; Section 5 shows the results of the proposed code compared to the original and alternatives, and Section 6 concludes the paper and describe the team future plan.

2. Literature Review

The purpose of using multiple search robots is to locate and collect any material found in a way that will optimize the ground covered efficiently. Most articles use algorithms to effectively and efficiently disperse their multiple search robots. The testing of algorithms or any other type of search-related coding is done through a simulation software in which a known environment is the home of resources a robot is seeking. The robots are programmed to follow a specific type of algorithm tend to follow a random direction path while systematically covering the complete area of the known environment. In addition, these multiple robots have been programmed to detect any obstacles in their paths and avoid collisions, making the probability of finding an object much higher.

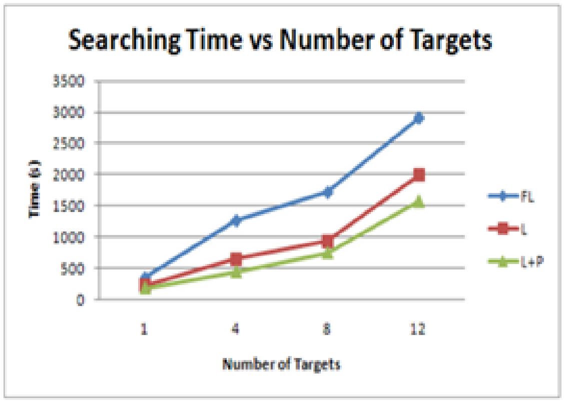

Figure1. *Searching Time vs Number of Targets*

The number of targets found compare to the time taken by three different algorithm approaches is shown in Figure (1), which are the levy random-walk and potential field (L+P) (green), levy random walk (L) (red), and the fixed-length random (FL) (blue). The levy algorithm generates the length of movement, while the artificial potential field improves the dispersion of the robots by generating a repulsion force [1].Figure (1) shows that the mixture of levy random-walk and the potential field (L+P) gathers the most targets in the shortest amount of time, meaning that the algorithm used to determine the length of movement and the dispersion amongst the robots results in finding the targets much quicker.

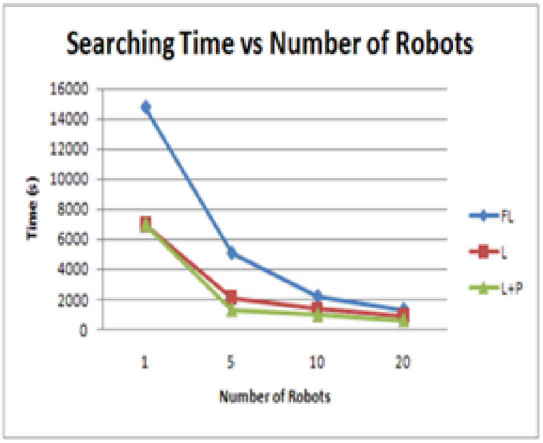

Figure2. *Searching Time vs Number of Robots*

Figure (2) displays the comparison between the three algorithms in terms of the number of robots used and how that will affect the time to seek and find a set number of targets. The results show that the levy random-walk and potential field (L+P) program will take the least time, but the other algorithms show they are close. The reason for this is that the potential field can have a negative effect when many robots are constantly changing paths since they are programmed to repulse against each other. DustySWARM has used the concept of both Levy random-walk and potential field while traveling in a reversed twister form to cover all necessary ground without missing any objects near the base location. In addition, the team programmed the robots to return and search nearby where an object has last been found. This is effective in scenarios where the objects are either clustered or semi-clustered.

3. Methodology & Algorithm Development

To incorporate the most adequate approach, the team brainstormed on different search patterns that would allow the robots to cover more area within the allotted time constraint. The objective of the competition is to collect the maximum amount of targets in a given area and return them to the base location. As knowledge and familiarity with the Robot Operating System (ROS) and the programming techniques increased, the code was modified to optimize target collection. The developed designs of searching algorithms were tested and compared with the default iAnt program and an updated iAnt program that included a “return logic” which allowed the rovers to return to a previous target location. This was done to allow the comparison of the same logic in a new search algorithm.

The team’s first design contained three rovers, and consisted of dividing the search area among the rovers, thus covering more area at a time. This approach had one roversearches the outer perimeter of the arena while the other two rovers searched the middle grounds. This design was soon discarded since all rovers must follow the same search algorithm.

The next design focused on scanning the arena in a square spiral pattern, sweeping from the outer perimeter of the arena to the center. This method brought an issue that the rovers were not able to continue with the square spiral pattern once they encountered a tag. Upon this event, the rovers would return to the center of the arena andcouldnot return to their previous location and orientation to continue with the pattern. This was mostly due to the random and cumulative errors of the IMU sensors. The resulting pattern of motion of this design seemed to approximate a spiral. This led us to the third design, which was a spiral motion.The Archimedean Spiral shown in figure (3) is the simplest of spirals and can be easily graphed using parametric equations [2]. The team decided that the rovers would have a more fluid pattern if they would follow such parametric equations to update the goalLocation variables. Hence, goalLocation.x and goalLocation.y would be updated with the values a*tcos(t) and a*tsin(t) respectively where (a) is a constant and (t) is the angle. One of the issues with this approach was (t) variable used to update the theta of the spiral. This (t) variable was treated as a time variable of the mobility step. In order to keep track of the current step of the mobility design, a global counter, mobilityCount, was used to count the number of times the mobilityStateMachine function was called. This counter variable was then used as the theta variable (t) of the design. It was then necessary, through trial and error, to adjust the (a) constant variable until the appropriate proportion of spiral was obtained.

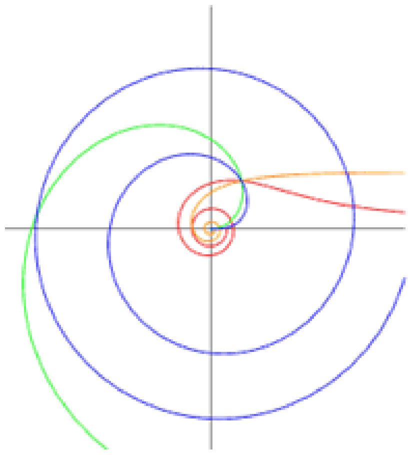

Figure3. *Archimedean Spiral*

Experimentation with the spiral algorithm yielded a value of 0.15 for the constant ‘a’. This value produced the best spiral for the rovers to navigate the arena. However, even with this optimum value, the team was not pleased with the spiral equation design since the rovers were constantly bumping into the outer perimeter of the arena.This impeded the rovers from searching the rest of the arena efficiently. Changing the ‘a’ constant did not seem to fix the issue since after a long enough time the rovers would still end up at simply bumping into walls because the mobilityCount would become large enough that the next point in the spiral pointed goalLocation to somewhere outside the arena.

The final design involved some overhauling of the spiral equations. A main change of the spiral equations was to make the rovers follow a clockwise rotation spiral, instead of counterclockwise.

Then the spiral equations were simplified to an approximation of a spiral. This was achieved by incrementing goalLocation.theta by -0.15 radians for the next target angle. Then goalLocation.x and goalLocation.y were updated with random scaleFactor, between 50 cm and 100 cm, multiplied by cos(goalLocation.theta) for x value, and sin(goalLocation.theta) for the y value of the goalLocation coordinates. This created a neat approximation of a spiral motion in a clockwise orientation. Thus, the final design was named Reverse-Twister Code. This was kept as the final design because of time limitations, but ultimately the team intends to continue improving this searching algorithm.

4. Experiment Simulation Runs

The iAnt uses a random search algorithm. It moves the rovers around by selecting a random heading angle, rotating the robot to the new desired theta value, and translating the robot for 50 cm in that direction. The result of this code is that the rovers look like they are lost and have no sense of orientation or purpose in their movement. Upon running the first couple of simulations, the team began to take note of what needed to be improved. Since the first couple of runs were done under the Clustered distribution, the team's primary goal was to develop code logic to have the rovers return to the previous target location after taking the collected tag to the collection disc. After scrutinizing the iAnt mobility file, a deeper understanding of the coding techniques was developed and the development of a customized code sprouted.

In order to verify the improvement of the search algorithm, the DustySWARM team tested the rover's collection capability while running the iAnt code. First, the team focused the evaluating the rovers to the iAnt code with the preliminary round settings. The iAnt performed poorly under Clustered target distribution but performed well under Power Law and Uniform distributions. The results of iAnt shown in Table (1)were used as a benchmark value that the team will outperform by designing their new search algorithm.

Table1.*Default iAnt Algorithm (Preliminary Runs)*

Target Distribution	Target collected
Clustered	15
Power Law	100
Uniform	124

The 'return logic' was implemented by saving the coordinates of the currentLocation upon detection of a target. These lines of code were placed in the targetHandler function in the body of the 'if statement' that verifies that a target has been detected. Later a global Boolean variable was created, 'returningToTarget', that indicated whether the rover had store coordinates of the previous location and was on its way to returning to the previous target location. This variable was used as a condition to decide whether the next goalLocation should be based on the search logic or the stored coordinates. This variable also helped to avoid the rovers losing the previous target location after encountering an obstacle while traveling to the previous location. After the obstacle-avoidance measures, the rover will use the mobilityStateMachine to decide what to do next, which will, in turn, use the variable that indicates the rover was returning to a target location, upon which the goalLocation was updated accordingly.

In future iterations, the default iAnt algorithm was modified to include the "return logic" that allowed the rover to return to a previous target location. The results of this modified iAnt algorithm are shown in Table (2). The results indicate an improvement under Clustered target distribution but had a small negative effect on the other two target distributions.

Table2.*iAnt Algorithm With Return Logic (Preliminary Runs)*

Target Distribution	Target collected
Clustered	33
Power Law	95
Uniform	110

It was assumed that other teams would probably be above this benchmark score since the iAnt algorithm was provided to all competitors as base code. Therefore, in order to have a competitive edge in this Swarmathon competition, the developed code must at least score above the default iAnt code.

As the code kept developing, new tables were created and were labeled as descriptive as possible so that the team may have control over which searching technique resulted in the optimal number of targets collected. The scores are shown in the following section, and they are the averages of repeated trials under each distribution for each search algorithm.

5. Results Summary

When the program ran first without any modifications in the coding, it was shown that within the Preliminary test, the swarm robots collected 15 targets for clustered distribution, 100 targets for Power Law distribution, and 124 targets for Uniform distribution, that was shown in Table (1). Modified iAnt with the return logic to the previous target location showed a new benchmark of what other teams might score if they only included a similar return logic. Resulted was shown in table (2), The team began trying different modifications and tests were run, the values the team was getting for Power Law and Clustered distribution were very low compared to the original code, as shown in tables (3-7). The team then began to tweak the operating linear and angular velocity. Each table below indicates a new change or a repeated trial with previously tested velocity and logic parameters. The first two tables tested the performance under Power Law distribution. Then the rest of the tables show the changes made and the performance for each target distribution.

Table3. *Twister Code with Velocity of 0.35m/secPreliminary Runs)*

Target Distribution	Target collected
Power Law	20

Table4.*Twister with modified (returningToTarget) handler, Velocity of 0.35m/sec(Preliminary Runs)*

Target Distribution	Target collected
Power Law	87

Table5.*Twister Code with original(mobilityCount), Velocity of 0.35m/sec(Preliminary Runs)*

Target Distribution	Target collected
Clustered	45

Table6. *Twister Code with modified (mobilityCount), Velocity of 0.35 m/sec (Preliminary Runs)*

Target Distribution	Target collected
Power Law	76
Clustered	63

Another modification was done to twister code, whichimplements an approximation of the spiral equation by using a scale factor of 0.25 times the distance to center from the currentLocation coordinates of the robot. The distance is found using Pythagorean theory equations where the hypotenuse is the distance to the center. This creates a vector of varying lengths, depending on the position of the robot in the arena. Thus, as the robot is further from the center, the next vector in direction goalLocation.theta will have a magnitude of 0.25*(sqrt(0.0-currentLocation.x)^2+(0.0-currentLocation.y)^2).

Table7.*Twister code with scaleFactor 0.25 and Velocity of 0.9 m/sec (Preliminary Runs)*

Target Distribution	Target collected
Clustered	73
Power Law	103
Uniform	114

Table 8.Shows the result of varying the scaleFactor variable from 0.5 to 1.5 meters instead of the magnitude of the tangent vector depending on the distance to the center. The result of this is a semi-random spiral that looks like an orbit around the collection disc.

Table8.*Random scaleFactor (0.5 to 1.5 m), Linear Velocity 0.8m/sec, and Angular Velocity of 0.3 rad/sec(Preliminary Runs)*

Target Distribution	Target collected
Clustered	70
Power Law	121
Uniform	146

Tables (9-12) show more tweaking of the twister code by changing the linear velocity using the algorithm of Table 8.

Table9. *Random scaleFactor (0.5 to 1.5 m), Linear Velocity 0.85m/sec, and Angular Velocity of 0.3 rad/sec (Preliminary Runs)*

Target Distribution	Target collected
Clustered	80
Power Law	101
Uniform	130

Table10. *Random scaleFactor (0.5 to 1.5 m), Linear Velocity 1.0m/sec, and Angular Velocity of 0.3 rad/sec (Preliminary Runs)*

Target Distribution	Target collected
Clustered	62
Power Law	104
Uniform	138

Table11.*Random scaleFactor (0.5 to 1.5 m), Linear Velocity 0.75m/sec, and Angular Velocity of 0.3 rad/sec (Preliminary Runs)*

Target Distribution	Target collected
Clustered	77
Power Law	102
Uniform	122
Uniform (Final Run)	247

Table12.*Random scaleFactor (0.5 to 1.5 m), Linear Velocity 1.5m/sec, and Angular Velocity of 0.2 rad/sec (Preliminary Runs)*

Target Distribution	Target collected
Clustered	73

Tables (13-14) indicate the results of the iAnt algorithm with modified return logic and then return logic with tweaks in the linear velocity. There was an improvement in changing the velocity to 1 m/s but 2 m/s was counterproductive since rovers were almost out of control. Table 15 is not complete because of this same reason.

Table13.*iAnt with Return Logic and Velocity 1.0 m/sec*

Target Distribution	Target collected
Clustered	58
Power Law	111
Uniform	118

Table14.*iAnt with Return Logic and Velocity 2.0*

Target Distribution	Target collected
Clustered	-
Power Law	90
Uniform	-

The team kept developing the twister algorithm in an attempt to surpass the scores of the modified iAnt search logic. The results of our default twister code are shown in table 15. With some modifications which included different velocity, scale factor, and angular velocity values and limitation to avoid setting the goalLocation setting to a location outside the perimeter of the arena for the final round, results had been improved and show in table 16.

Table15. *Default twister code*

Target Distribution	Target collected
Clustered	57
Power Law	71
Uniform	83

Table16. *Updated TwisterscaleFactor (1.0 m), Linear Velocity 0.8 m/sec, and Angular Velocity of 0.2 rad/sec (Preliminary Runs)*

Target Distribution	Target collected
Clustered	58
Power Law	113
Uniform	122

Striving for improvement, the team kept tweaking and figuring more variables that they can control to improve the code performance. Rovers will be moving in a clockwise rotation around the collection disk with an approximation of a spiral while limiting the perimeter logic. The Spin Test Factor selected a random number that indicates the length of the tangent vector used to approximate the spiral. With this range of vector length, the rovers were able to orbit the collection disk in an elliptical spiral, improving the number of tags collected significantly. The results of these modifications are illustrated in the Tables (17-19)

Table17. *Updated Twister scaleFactor (0.2-0.75 m), Linear Velocity 0.8 m/sec, and Angular Velocity of 0.2 rad/sec (Preliminary Runs)*

Target Distribution	Target collected
Clustered	78
Power Law	110
Uniform	117

Table18. *Updated Twister scaleFactor (0.5-1.5 m), Linear Velocity 0.8 m/sec, and Angular Velocity of 0.3 rad/sec (Preliminary Runs)*

Target Distribution	Target collected
Clustered	97
Power Law	135
Uniform	157

Table19. *Updated Twister Final Round Arena with limit parametersscaleFactor (0.5-1.5 m), Linear Velocity 0.8 m/sec, and Angular Velocity of 0.3 rad/sec*

Target Distribution	Target collected
Clustered	134
Power Law	195

6. Conclusions and Future Work

The twister code developed by the DustySWARM robotics team demonstrated to be superior to the default iAnt-searching algorithm. Many changes and tweaks were made to the twister code in order to keep improving the number of targets collected. The team achieved the desired target collection outcome after continuous modifications and testing of the algorithm developed. Despite the increased efficiency obtained with the reverse twister code with random scaling, the team agreed that there is still room for improvement since it should be possible to collect almost all, if not all, the tags.

Therefore, the team plans to continue their collaboration with the NASA Swarmathon team and will focus on developing an even better search algorithm within the next year, for the following Swarmathon Competition. Some of the improvements to be implemented is to have communication between the rovers; when there is a large pile of targets detected, the rovers should collaborate to collect all the targets of the discovered cluster. Another improvement is to implement a method of accounting for drift from the encoders. These implementations will require learning about the ROS (Robot Operation System) libraries and functions.

DustySWARM team 1.0with their Reverse-Twiter Search algorithm placed the third in the virtual competition among twelve (12) participating teams from all over the USA.

Acknowledgment

Thanks to NASA and the University of New Mexico (UNM) teams for all the help and the opportunity to participate in such great competition. Thanks to all our sponsors from Texas A&M International University (TAMIU) and Laredo.

References

[1] Donny K. Sutantyo, Serge Kernbach, Valentin A. Nepomnyashchikh, and Paul Levi: Multi-Robot Searching Algorithm Using Levy Flight and Artificial Potential Field, Mountain View, CA:University of Stuttgart, Universitatstrasse 38, Stuttgart, Germany, August 2011

[2] https://mathworld.wolfram.com/ArchimedeanSpiral.html

[3] W. E. Shots, "The Linux Command Line", 2nd ed., San Francisco, CA: Creative Commons, 2013.

[4] J. M. O'Kane, "A Gentle Introduction to ROS", Columbia, SC: University of South Carolina, 2014.

[5] http://nasaswarmathon.com/

AUTHOR'S BIOGRAPHY

Dr. T. Tashtoushis an Assistant Professor at the School of Engineering in Texas A&M International University (TAMIU), Laredo, TX. He got his Ph.D. and M.S. degrees in Systems and Industrial Engineering from State University of New York (SUNY) at Binghamton on 2013 and 2009, respectively and his B.S. in Mechatronics - Mechanical Engineering from Jordan University of Science and Technology (JUST), Irbid, Jordan on 2005. He is a multi-discipline engineer, who has industrial and academic experience in the field of Systems Simulation and Design, Production Quality and Management, Lean Manufacturing, Six-Sigma Processes, Industrial Robotics and Automation, Exploration and Autonomous Robotics, Artificial Intelligent (AI), Computer Integrated Manufacturing, 3D Printing Processes, Engineering Statistical Analysis, Project Management, Optimization, Instruments and Electrical Devices, Reliability, Healthcare Systems, Nano-Technology and Energy Harvesting, and Human Factors/Egomaniacs Studies.

Citation:*Tariq H. Tashtoushet.al., (2020). "Reverse-Twister Swarm Search Algorithm Design: NASA Swarmathon Competition", International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), 7(1), pp.29-36.DOI:http://dx.doi.org/10.20431/2349-4859.0701004*