NASA Swarmathon
Team ABC (Artificial Bee Colony)

Cheylianie Rivera Maldonado, Kevin Rolón Domena, José Peña Pérez,
Aníbal Robles, Jonathan Oquendo, Javier Olmo Martínez

University of Puerto Rico at Arecibo
Dr. Eliana Valenzuela-Andrade

Abstract

The NASA Swarmathon is a competition that challenges Minority Serving Institutions to develop integrated robotic platforms that improve extraplanetary resource retrieval rates by using swarm robots. Our team's goal was to research efficient search algorithms for the virtual swarm robots using C++, Robot Operating System (ROS), and Gazebo. Our robots must be able to autonomously search and retrieve resources scattered across an arena and return them to a collection zone in a simulated environment. Our methodology was to implement our own version of the Artificial Bee Colony (ABC) algorithm by Derbis Karaboga. We decided to use Karaboga's algorithm because its performance excelled other's swarm intelligence and population based algorithms. In the ABC algorithm, the colony of artificial bees contains three groups of bees: *onlookers*, *employed* and *scouts*, but our team's implementation only needs the scouts and the onlookers. We need the robot scouts to search for clusters of resources and to call the onlookers when they find them. The onlookers will then proceed to retrieve all the resources in the cluster that was found. With this approach we hope to exploit the clustered distribution of resources and collect a high amount of them. This semester, we could accomplish an almost finished version of the ABC algorithm, with a successful implementation of the onlookers and the scouts. We earned an 8th place in the NASA Swarmathon competition, against twenty-three Minority Serving Institutions and Community Colleges across the United States and Puerto Rico.

Introduction

The NASA Swarmathon challenges Minority Serving Institutions to develop integrated robotic platforms that improve resource retrieval rates by using swarm robots; that could be used in space exploration. Swarm Robotics is the study of how to design groups of robots that operate without relying on any external infrastructure or on any form of centralized control. The design of swarm robots is guided by swarm intelligence principles. Swarm Intelligence refers to sophisticated collective behavior that can merge from the combination of many simple individuals, each operating autonomously (independently).

Computer Science students of the University of Puerto Rico at Arecibo (UPRA), will be part of the national simulated competition of NASA, Swarmathon. The simulated competition will not require the use of the actual swarmie hardware. Swarmie, also known as autonomous robots as mentioned in the first paragraph, are intelligent machines capable of performing tasks by themselves, without explicit human control. The essential characteristic of swarm intelligence consist of biological inspiration; understanding of the decentralized mechanism that underlie the organization of natural swarms such as ants, bees, birds, fish, wolves, and even humans.

For the competition, teams will need to modify the code provided by NASA, implement a search algorithm and test the modified code with three types of resource distribution: uniform, clustered and power law. The goal of the simulated competition is for the virtual swarmies to be able to search in a square arena and maximize or collect a significant amount of resources during a specified time, based in two competition rounds: preliminary and final.

UPRA students chose one of the most used swarm algorithms, known as Artificial Bee Colony (ABC). Before choosing ABC, a search pool of algorithms was held to evaluate and determine which of the following algorithms will be implemented for the competition: Artificial Bee Colony (ABC), Firefly Algorithm (FFA) or Particle Swarm Optimization (PSO).

Related Work

Our team's methodology on NASA's very first Swarmathon was to implement the Artificial Bee Colony algorithm described in Derbis Karboga's 2007 article, *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm* [1]. We decided to use Karaboga's algorithm because its performance excelled other's swarm intelligence and population based algorithms. The algorithms that were tested alongside the ABC were the following: Particle Swarm Inspired Evolutionary Algorithm (PS-EA), Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Each algorithm was tested with five high dimensional numerical benchmark functions that have multimodality. Benchmark functions are *test functions for optimization*. The five test functions used in these experiments were: Griewank function, Rastrigin function, Rosenbrock function, Ackley function and Schwefel function. The best solutions found by the algorithms were recorded. The recorded results indicate that while the performance of PS-EA algorithm deteriorates in optimizing difficult functions like Griewank and Ackley functions, the ABC algorithm shows better performance on them. Each experiment was repeated 30 times with different random seeds. From the simulation results it was concluded that the ABC algorithm has the ability to get out of a local minimum and can be efficiently used for multivariable, multimodal function optimization.

In the ABC algorithm, the colony of artificial bees contains three groups of bees: *onlookers*, *employed* and *scouts*. A bee waiting on the dance area waiting for a decision to be made on choosing a food source, is called an onlooker. A bee carrying out a random search is called a scout and a bee going to the food source previously visited by itself is named an employed bee. For every food source, there is only one employed bee. In other words, the number of employed bees is equal to the number of food sources around the hive. The employed bee whose food source is exhausted becomes a scout. These are the main steps of Karaboga's ABC algorithm:

• *Initialize.*

• *REPEAT:*

(a) Place the employed bees on the food sources in the memory;

(b) Place the onlooker bees on the food sources in the memory;

(c) Send the scouts to the search area for discovering new food sources.

• *UNTIL (requirements are met)*

In the ABC algorithm, each cycle of the search consists of three stages. At the **initialization stage**, a set of food source positions are randomly selected by the bees and their nectar amounts are determined. Then, these bees come into the hive and share the nectar information of the sources with the bees waiting on the dance area within the hive (the onlookers). **At the second stage,** after sharing the information, every employed bee goes to the food source area visited by herself at the previous cycle since that food source exists in her memory and chooses a new food source based on visual information. Visual information is based on the comparison of food source positions. When the nectar of a food source is abandoned by the bees, a new food source is randomly determined by a scout bee and replaced with the abandoned one. At the **third stage** In order for a food source to be exhausted by the onlooker bee, the food source has to be rich on nectar amount.

How do bees communicate this information to other bees?

Bees communicate with each other through dancing. The dance occurs on a special dance floor, which is conveniently located near the entrance to facilitate quick entry and exit of foragers, and only bees with news of highly profitable sources of nectar execute the dance. Arriving back at the nest, a bee with news to share immediately proceeds to the dance floor, where other bees waiting for news gather around her.

The bee dance takes two distinct forms, depending on the distance of the food source. The form known as the “round dance,” encountered most frequently, doesn’t bother to indicate the food source’s distance and direction. It does, however, tell the workers that the source is closer than 15 meters (50 feet) from the nest. The other form is called the “waggle dance”, during which she dances a figure-eight pattern, with a straight "walk" in between the loops and a sporadic fluttering of her wings.

The worker communicates several key pieces of information during the dance. The longer she waggles - typically bees make between one and 100 waggle runs per dance - the farther the flower patch lies from the hive, with every 75 milliseconds she prolongs the dance adding roughly another 330 feet to the distance. She shows how rich the source is by how long and/or how vigorously she dances. Perhaps most astonishingly, she indicates the direction of the source by the angle her waggle walk deviates from an imaginary straight line drawn from the dance floor to the sun at its current position. The onlooker bees watch this dance and decide which source food they will go to.

Other algorithms were considered, like the Firefly Algorithm, but ultimately the ABC algorithm was chosen as our preferred method of execution. Our work builds on the ABC algorithm by visualizing our simulated robots as a bee hive, and implementing some of Karaboga’s most important concepts in a way that worked for us.

Methods

ABC Algorithm Method: Our implementation

Our team's implementation of the ABC algorithm concentrates on two groups of bees, the scouts and the onlookers. We don't use the employed bees type because we don't know where our food sources (tags) are going to be and we are limited to only three and six bees per simulation. The team's approach on Karaboga's algorithm begins with one onlooker, bee (rover), and two scout bees on preliminaries and four scout bees and two onlooker bees on finals. When the simulation begins, the scout bees begin to randomly explore their designated quadrants, I and II for one scout, III and IV for the other scout. In finals there are two scout bees exploring quadrants I and II, and two scout bees exploring quadrants III and IV. When a scout detects a food source rich in nectar (a cluster of tags), it calls the onlooker bee to the cluster's position. As the third stage of Karaboga's algorithm states, the onlooker will only go to a food source if it's rich on nectar. Once it reaches the cluster's position, it will start to collect the tags and bring them to the collection zone. Our onlookers will wait for three minutes to see if it's called to visit a cluster. If it's not called, it will also start to randomly scout the area. With this methodology and with room to always improve, we expect to collect as many tags as we can.

In order to successfully design and implement our algorithm we had to work with the programming language C++, alongside the "Robot Operating System" (ROS) framework. ROS is a set of tools that helps you build robots application. It's a system that has a complex hierarchy. Viewing ROS from a high perspective, it's composed of a "ROS Master", which allows all others pieces of ROS (nodes) to transfer information between them. Nodes communicate with others nodes using a concept of "publish" and "subscribe" method, in other words, they share information with each other. We also utilized a robotic simulation software called "Gazebo", in order to simulated the populations of robots in a complex environment.

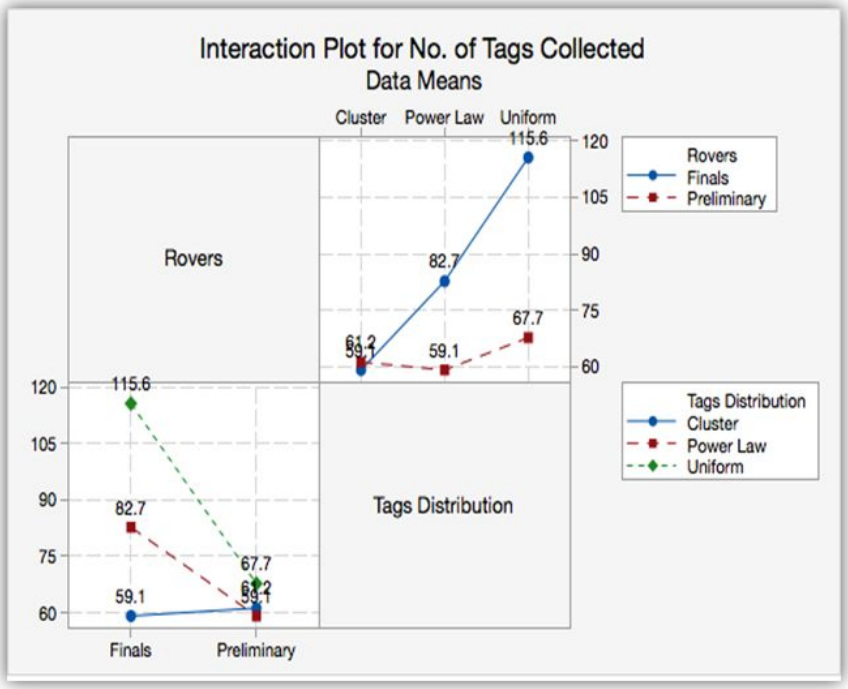
Our algorithm:

```
Initialize rovers' positions near the center of the arena
Send scouts to look for resources at specific quadrants
Keep onlookers at the center until called
REPEAT
      if a scout finds a resource
          if resource is part of a cluster
              Give coordinates of the center of arena to scout
               Send scout to the center of the arena to drop off
resource
              Push the coordinate of the cluster to the cluster queue
               Give coordinates of first cluster in queue to the
            onlookers
              Send the onlookers to the resource
          else
               Give coordinates of the center of arena to scout
                Send scout to the center of the arena to drop off
resource
      if an onlooker finds a resource
          Give the coordinates of the center of the arena to onlooker
            Send onlooker to the center of the arena to drop off
resource
      else
          Keep scouts looking for resources
          Keep onlookers at the center until called
UNTIL (all resources are collected)
```

Experiments

We designed an experiment with the purpose to investigate the performance of our algorithm, using the different tags layouts (cluster, uniform, power law) on the two categories provided, “Preliminary” composed of 3 robots, and “Final” with 6. At the beginning we start with the rovers in completely random search . Then we compare the results with specific rovers searching by specific quadrants (semi-random search) in order to optimize the searching phase. The last one was the most efficient of the two. For each of our improvements we ran 10 times each combination, controlled the time of each simulation and recorded the number of tags collected by the rovers. Two way ANOVA were used for testing the data obtained from the study using 0.05 level of significance. Each time the team manage to improve in some way the code, we had to recompile our values to create a new statistic form, in order to see if there was any difference with our implementation. Currently, the compile data does not show a significant improvement to the original code provided by Nasa, although, our algorithm still has much room for improvement.

Results


Interaction Plot for No. of Tags Collected
Data Means
Cluster
Power Law
Uniform
Rovers
Finals
Preliminary
Tags Distribution
Cluster
Power Law
Uniform
Rovers
Tags Distribution
120
105
90
75
60
115.6
82.7
67.7
59.1
61.2
Finals
Preliminary

Conclusion

To do a successful performance during this research experience, we had to learn how to program using C++ programming language and the Robot Operating System (ROS), along with its simulator, Gazebo. We researched different biological algorithms and decided to choose Derbis Karboga's Artificial Bee Colony algorithm as our inspiration. We adapted it to fit our needs, modifying it in ways that would work for our swarm robots. We successfully learned to work with ROS, a tool previously unknown to us, and developed functioning code that awarded us an 8th place at the Swarmathon Virtual Competition.

Although the initial high learning curve slowed our progress at first, we are now fully capable and motivated to continue to improve our algorithm so that it can function at its full potential. Functions involving the onlooker retrieval behavior will be revisited in order to fully exploit the cluster distribution. As part of our future work plans, we have scheduled to replace our queue with another data structure, like a linked list, that can allow us to change the positions of the clusters that have been found according to their level of fitness, so that onlookers can go to the best cluster available. Fitness will be based on the size of the cluster, and more importantly, on the distance from the cluster to the center. We will also create a small area around the cluster coordinate for the onlooker to explore, so that it will have more chances of emptying that cluster, instead of returning to the exact same position to look for more tags.

References

[1]Karaboga, D. and Basturk, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization 39*, 3 (2007), 459-471.

[2]NOVA Online | Tales from the Hive | Communication. *Pbs.org*, 2016. <http://www.pbs.org/wgbh/nova/bees/hivecomm.html>.

[3]"Nasa Swarmathon Competition." *Nasa Swarmathon*. N.p., 1 October 2015. Web. 20 April 2016. <http://nasaswarmathon.com>.

[4]"Documentation." *Ros.org*. N.p., N.p., Web. 10 May 2016. <http://wiki.ros.org>.

[5]O'Kane, Jason. *A gentle introduction to ROS*. Columbia: CreateSpace Independent Publishing Platform. 2016. Print.

[6]Joseph, L. *Mastering ROS for Robotics Programming*. Birmingham. Packt Publishing. 2015. Print.