Implementation of the Central Place Foraging Algorithm With More Realism

Valuable Sheffey
School of Engineering
University of New Mexico
Email: vivieltwixt@gmail.com

Dr. Melanie Moses
School of Engineering
University of New Mexico
Email: melanie.moses@gmail.com

Dr. Matthew Fricke
School of Engineering
University of New Mexico
Email: matthew@fricke.co.uk

***Keywords*—ARGoS, iAnts, Gazebo, ROS, foraging algorithm, swarm robotics**

I. Introduction

Effectively collecting dispersed resources and transporting them to a specific location is a complex task with many useful applications. This behavior, called foraging, has been replicated in the central-place foraging algorithm (CPFA) [1] created by Moses and Hecker to run on robot swarms. CPFA is an adjustable strategy for foraging in environments of variable resource distributions, various swarm sizes, and a range of machine error. It is inspired by desert harvester ants which use site fidelity and pheromone trails in their foraging strategy. CPFA was implemented in simulation and in physical robots by Moses and Hecker in 2015 using ARGoS and iAnt robots. The researchers were able to to measure the efficiency of the algorithm in both environments in terms of error tolerance, flexibility, and scalability.

The use of CPFA in real-world applications requires more finesse than the abstraction provided by the lab environment of Moses and Heckers 2015 experiments. For example, in applications such as space exploration, a robot retrieving a resource may face the challenge of actually picking up the resource versus simply scanning it (as was done in the aforementioned experiment). For this reason, we seek to understand how CPFA performs on robots within a more complex framework. Understanding CPFA performance in more realistic settings can help us better predict performance in real world applications. This knowledge can help determine the appropriateness of applying CPFA to specific situations.

II. Background

The previous experiment on which we build our current research is the experiment in Moses and Hecker 2015, *Beyond Pheromones: Error-Tolerant, Flexible, and Scalable Robot Swarms*. In that experiment, the researchers developed the foraging algorithm CPFA and modelled localization and tag detection sensor error. Then, they used a genetic algorithm to evolve the algorithms parameters for particular experimental conditions. Finally, they performed simulated and physical experiments to measure the algorithms efficiency in these different environments.

A. CPFA algorithm

This foraging algorithm is inspired by desert harvester ants which use site fidelity and pheromone trails in their foraging strategy. Site fidelity is the tendency to return to a site where a resource was found. Pheromone trails are pathways laid by ants to lead other ants to a site of high density in resources.

B. Sensor Error Modelling

The iAnts had significant sensor error in localization and tag detection. To produce more realism in the simulation, the researchers seeked to model this error. They modelled localization error via linear regression of measurements of standard deviation error on the rovers' reported distance from the central nest. For tag detection, they determined the percentage of tags the rovers were able to correctly identify.

C. Genetic Algorithm

A genetic algorithm was used evolve the algorithms parameters for specific swarm sizes, resource distributions, and error models. Fitness was measured as foraging efficiency, the number of resources collected by all rovers per run.

D. Experiment

To measure CPFA performance, the researchers performed both physical and simulated experiments.

1) Physical: Each experiment ran for 1 hour outdoors on a 100 m^2 concrete surface. The central collection zone was represented as the area around a beacon. Resources were QR tags. Resource collection was simulated by rovers scanning a found tag. Dropping off a resource involved approaching the beacon and reporting the tags info.

2) Simulation: Simulated experiments were similar to physical ones. Robots searched a 100 m^2 area and each run was 1 simulated hour. The simulations were run with a program called ARGOs in which the iAnts actual dimensions, travel and search speeds, and resource detection areas were used. Some simulations used the error model to replicate sensor errors.

Performance was measured by efficiency (the number of resources collected by all robots per run) in terms of error tolerance (ability to adapt to robot error), flexibility (ability to perform well on distributions the algorithm was not optimized for), and scalability (ability to increase efficiency as number of robots increase).

III. Methods

A. Substitutions

We intended to replicate most of Moses and Heckers work with an added layer of complexity. We implemented the CPFA algorithm in ROS. The ARGoS simulator was replaced by Gazebo. Instead of iAnts we used their upgrade: rovers.

1) ROS: ROS, the Robot Operating System, is an open-source project to make developing robotics software easier. A community-driven initiative, ROS lets experts in particular areas (motion planning, etc.) create and share their packages. ROS's emphasis on modularity allows a user to choose which packages are useful for his/her work. This collaboration improves robotics as a field overall, as users build upon each others work. The previous experiment had not taken advantage of these features, so we chose to do so here.

2) Gazebo vs ARGoS: For the purposes of this experiment, we preferred Gazebo to ARGoS. Compared to the abstraction model given by ARGoS, Gazebo provides more realism. For example, Gazebo has more realistic physics. Gazebo also provides a more nuanced interface, allowing better control over the simulations. Most importantly however, Gazebo interfaces with ROS in a way that allows us to use the same ROS code for both the simulation and the real rovers. This is very useful in that it prevents us from writing the same code twice.

3) Rovers vs iAnts: Rovers are the upgraded version of iAnts. Features such as increased robustness, added gripper functionality, GPS tracking, and the ability to add more hardware bring these robots closer to what will be their ideal form in real-world applications.

B. Implementation

To implement CPFA in ROS and Gazebo, we chose to use code from the Swarmathon challenge as our basis. With this in mind, we studied in depth the algorithm of Moses and Hecker's paper. When we gained adequate understanding, we reviewed the actual code of their ARGoS implementation. Using both as a guide, we developed a pseudocode for implementing the algorithm in ROS. We broke down the pseudocode even further into discrete tasks that could be checked off. We then peer coded until all the tasks were completed.

Codewise, this meant creating a CPFASearchController class within the mobility package (package responsible for rover movement). The highlight of this class was the CPFA state machine that modelled the state transitions of the algorithm. We also coded a separate class to represent the pheromones.

We chose to peer code because design decisions had become so interdependent that it had become unproductive to code separately due to the constant consulting required.

After this, we debugged to ensure the algorithm ran as expected. For me personally, the most frustrating bug was the rovers' inability to correctly choose a pheromone. That is, whenever a rover decided to follow a pheromone laid by another rover, instead of going to the pheromone site, it would go to its previous target site. I eventually narrowed down the error to the update function which is called before selecting a pheromone. This code automatically makes all pheromones inactive, so the rover has no options to select from. This explains why the rover exits the pheromone selection function without changing its previous target site. The update function appeared to have been implemented exactly as in ARGoS. In fact, the code was near identical as both were coded in C++. The only difference was that one used an ARGoS time function and the other a ROS. I became convinced that this was the issue and scoured the web for documentation on both functions so I could compare them. When I finally found the documentation, it turned out that both functions returned the same value using the same units. This stumped me as it meant the code was correct yet it was certainly responsible for causing all the pheromones to become inactive before they could be selected.

After taking a break with other tasks, such as Hackathon preparation, I finally figured it out. The code was implemented fine. The issue was that we had set the default parameter for the pheromone decay too high. Just before a rover selects a pheromone, it updates the pheromones list using the decay function which (due to the high decay rate) automatically inactivates all pheromones. With some tweaking, I was able to reduce the decay rate to a reasonable rate.

Other errors were of similar nature - could be fixed by changing a default parameter - or straightforward goofs that we were able to resolve quickly.

As described, we implemented CPFA using ROS on rovers in Gazebo. We did not reimplement the genetic algorithm. Parameters for the CPFA were chosen manually, that is, parameters were selected based on what appeared to work well for the given environment and swarm size. We also forgoed sensor-error modelling.

IV. Discussion

The magnitude of this project means that much more work remains for the current and future REU students. Future work includes modeling rover sensor error and integrating it into the simulation, using a genetic algorithm to more scientifically determine the best parameters for each of the environments, implementing CPFA on the physical rovers, and performing experimental runs to determine the effectiveness of the parameters chosen.

V. Addendum: RSS Swarmathon Hackathon

The RSS Swarmathon Hackathon is an 18 hour challenge in which participants work in teams to code rovers to autonomously collect blocks and return them to a central collecting zone. In addition to CPFA research, my DREU experience included assisting in Swarmathon Hackathon preparations and serving as an advisor to participants.

My main role in Hackathon preparation was to participate in mock hackathons to test the difficulty of the tasks anticipated for participants. In the first mock hackathon, me and my team were given very bare bones code. We needed to implement a lot of missing functionality such as the ability to pick up blocks, to avoid obstacles well, and to localize. Unlike the usual Swarmathon competition in which rovers could use GPS, these rovers would need to localize using info from cameras that reported rover and resource locations using their tags. Localization was the hardest part. We were able to get the rovers to move reliably, to receive and move to a tag location, and to pick up blocks. We were working on the algorithm for parceling the tags to each rover, improving obstacle handling, and fixing bugs in localization when we ran out of time. In the second hackathon, we had a better idea of how to approach the challenge. The first mock hackathon had shown that expecting the teams to implement effective localization in addition to other tasks was not reasonable in the time allotted. In the second hackathon we worked from code with localization already implemented. Unfortunately the code had some errors, so we spent most of our time working with our mentor to resolve these issues. Because of this we made progress, but did not complete all the tasks, though we came close. These mock hackathons provided our mentors with enough information to revise and finalize the code for the actual hackathon.

At the actual hackathon, we REU students were designated as advisors. We rotated about every 1.5 hours between teams. I was nervous that I wouldn't know enough to be useful, especially since I had never participated in the actual Swarmathon and many of the participants had. However, the experience I had gained from the mock hackathons and internship in general allowed me to provide a lot of helpful advice to the participants. I was able to explain ROS concepts, the general layout of the code, advice on time management, and more. Whenever a problem came up that I couldn't solve, I simply referred to another mentor for guidance.

Though only one team managed to retrieve any blocks in the end, all the teams were amazing. Some teams were just beginning to learn ROS basics at the Hackathon and still made significant progress. Knowing how challenging their task was, I was inspired to see how much these students achieved. I am also inspired by how much I've too learned and achieved over these few months.

ACKNOWLEDGMENT

I would like to thank Moses Biological Computation Laboratory for its guidance and mentorship. Also, special thanks to Distributed Research Experiences for Undergraduates (DREU) for making this research possible.

REFERENCES

[1] Hecker, Joshua P., and Melanie E. Moses. "Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms." *Swarm Intelligence* 9.1 (2015): 43-70.