How effective is Ant Colony Optimisation at Robot Path Planning

Aaron Wolfenden[1], **Neil Vaughan**[1]

Abstract. This project involves investigation of the problem robot path planning using ant colony optimisation heuristics to construct the quickest path from the starting point to the end. The project has developed a simulation that successfully simulates as well as demonstrates visually through a graphical user interface, robot path planning using ant colony optimisation. The simulation shows an ability to traverse an unknown environment from a start point to an end and successfully construct a route for others to follow both when the terrain is dynamic and static.

1 INTRODUCTION

To be able to prove that a solution to the problem was found and that a good level of success had been achieved, the graphical user interface (GUI) displays, the pheromone trail and the obstacles that occurred which the agents rerouted around, using a grid based output shows the path the agents are taking throughout the solution. This was comparable to other examples of solving the solution with ant colony optimisation (ACO) to see if improvements have been made and whether some ACO variations are better than others depending on the environment.

Robot path planning (RPP) is used within many applications within the technological world, from helping unmanned vehicles to robotics. It is stated by (Cao, 2016) as ‘to find a path from the current point (or the start point) to the target point, which is a shortest or a minimum price path without barrier’.

ACO is a heuristic based off the foraging behaviour of ants, originally created by Marco Dorigo and in 1991, and further expanded on in later years. It was originally the ‘Ant System’ as a basic heuristic for solving optimisation problems such as the Travelling salesman problem (TSP). However later became developed further and more optimised, becoming ACO.

Within the field of artificial intelligence (AI), ACO is becoming increasingly useful. This is especially reinforced when Google, Uber and other companies are experimenting and testing the idea of self-driving cars and thus many career paths are becoming open in the field.

2 BACKGROUND

A literature search was conducted, the scope of which includes the different techniques that have been employed historically to achieve ACO and different variants and modifications that have been made to the algorithm to optimise it in most part to solve the TSP. Furthermore, investigation was made into the area of RPP and the implementation methods that were used to solve this issue.

As well as this, other areas of ACO were investigated such as the usage of artificial neural networks and genetic algorithm hybrid heuristics.

There are some solutions already applied to this problem, such as the artificial potential field method, neural networks, genetic algorithms (GA) and A* searching method as stated by (Yu, Wei, Wang, Ding, & Wang, 2017). However, they went on to state that each of the current solutions does have problems associated to them. There has also been attempts at hybrid models too consisting of different swarm intelligences integrated together.

There have been many different solutions created to try tackle this problem such as genetic algorithms which is the metaheuristic focussing on evolutionary algorithms that are inspired by natural selection and particle swarm optimization which iteratively searches for improvements to a candidate solution. However, the Swarm Intelligence this paper focussed on is ACO which although has proven to work effectively in solving the problem, still has improvements that could be made and investigated.

Another variation of ACO that implements GA into it is smartPath, created by (Châari, Koubâa, Bennaceur, Trigui, & Al-Shalfan, 2012). Within their work they proposed system they state that ‘ACO has a stronger local search capability and faster convergence but the algorithm can easily sink into a local optimum’ where as GA ‘belongs to random optimize processes, so the local convergence problem does not appear; however, this makes its convergence speed slower’. This therefore clearly shows that the shortcomings of one can be improved by the other.

The way they therefore designed the algorithm was so that the initial path is created from using ACO to create a fast-converged optimal path, and the second phase which is using the GA as a way of post optimization to improve the quality of the solution. This is done by checking all the nodes and then attempting to mutate the nodes in the path if the length of a resulting new path is shorter.

When tested against other heuristics such as their improved ACO (IACO) and classic ACO (CACO) and even GA it was found to outperform each of them in a varying number of environments in terms of both finding the shortest path and the efficiency with time.

Another variation on ACO is AntFarm designed by (Collins & Jefferson, 1990). Their idea was to create a simulation of an evolving population of ant colonies, where the reproduction is based on the amount of food they can carry back to nest, thus promoting better foraging strategies. The colonies are made up of identical ants, however their behaviour is specified by an artificial neural network (ANN).

[1] Dept. of Computer Science, Univ. of Chester, CH65 7AL, UK. Email: {n.vaughan}@chester.ac.uk.

This approach is interesting as it implements a genetic algorithm (GA) within the ACO algorithm. With this they aimed to implement natural evolution using local competition and mating. The other main interest within this research is the use of an ANN to represent each ant's behaviour. Through their use of GA, they can mutate the ANN allowing evolution of each colony of ants. The advantage of a technique such as this is that they can mutate and optimise many features of the ants such as pheromone density, how the optimal path is defined and other factors.

3 HEURISTIC DEVELOPMENTS

The way in which the heuristic functions is that the agents (ants) will go out from the start point (nest) and search for the end objective (food) when the ant reaches the food it will return to the nest and lay a track of pheromone. Until the path to the food is discovered the ants will decide on random directions, without any decision making. However, when a food source is discovered and they pick up a pheromone trail is increases the probability that future ants will then take this route. Overtime this trail will strengthen and most agents will follow this trail.

As well as being able to handle single-objective optimization problems where only one food source is available, (Dehuri, Ghosh, & Cho, 2011) states that ACO also excels at multi-objective optimization problems. These are problems where there may be conflicting objectives which solve the same solution. So, two food sources of equal value, and many ways are presented to solve this issue in their literature review.

Improvements of ACO are also becoming more common as (Yu, Wei, Wang, Ding, & Wang, 2017) states the advantages of ACO are 'its strong robustness' and more importantly that it's 'easy combination with other algorithms'. It was also investigated how much further it would optimise ACO with the addition of evolution within the algorithm brought over from GA. (Roach & Menezes, 2008) state this also, putting forward the evolutionary ant colony optimization (EACO). They further state that through using EACO, it's possible to 'give the individual ants a chance to evolve, and thus, the agents themselves can become more optimized' and they further show in their paper that it is much more powerful in dynamic environments than ACO however ACO is more efficient in static environments.

4 ROBOT PATH PLANNING ACO METHODS

Cao (2016) puts forward an improved version of ACO for RPP. Within it they stated that the initial search time takes too long. This is the result of the randomness at which the ants initially search. Therefore (Cao, 2016) uses a pheromone which decreases with distance with the idea that the ant will have a clear motion direction during the initial search.

Another criticism that (Cao, 2016) made is that the pheromone evaporation rate is unchanged throughout the running of the algorithm, which can lead to local convergence if too small, or slow convergence rate if too large. Therefore, they put forward the notion that a dynamic evaporation rate should be used. This is done with setting the evaporation rate high at the start to enhance the global search ability, then lowering it with the number of cycles so that local convergence can happen on an optimal solution quickly. (Yu, Wei, Wang, Ding, & Wang, 2017) also criticise the original ACO for this and suggest an 'adaptive pheromone volatilization coefficient' where they again suggest that evaporation rate should change with the number of cycles.

The heuristic function in previous work was improved upon by (Cao, 2016). This is done by changing the heuristic function so that the distance between two grid points which is normally used is changed so that instead what is used is the distance between the next grid point the ant will move to and the target point if it is known. This idea is also supported by (Yu, Wei, Wang, Ding, & Wang, 2017) where they also improve the heuristic so that the point that is closer to the goal is chosen instead of blindly picking. The advantage of this is that the initial search speed will be greatly improved and give the agents some direction when searching out a path.

However, (Cao, 2016) uses the ant-cycle algorithm described by (Dorigo, Maniezzo, & Colorni, 1996) after a cycle of movements made, optimal and worst solutions are calculated and some of the best solutions are used and the pheromone quantity is updated as a result.

5 IMPLEMENTATION

The methods aimed to optimise the ACO algorithm as much as possible, so the initial work focussed on an approach to firstly work on the core system methods and then expand to optimising performance, and creating a more complex GUI to control parameters.

The system was created using the python programming language. The reasoning behind this choice was the ease to develop in a short space of time with its extensive library support, readable code and finally due to it being interpreted it is much easier to debug. Libraries used within the creation of the program include Matplotlib which aided in the creation of the graph / grid interface. Another package used is the Tkinter package which provides the GUI with the controls to modify the functionality of the simulator, and was the base GUI for the graph to be imbedded within.

The project was also designed within the Pycharm IDE created by Jetbrains. The reasoning behind this decision was that the IDE provided all the tools and functionality required, especially ease of package management and syntax correction.

The ideas presented in this paper have been successfully implemented as a proof-of-concept prototype as shown in Fig. 1.

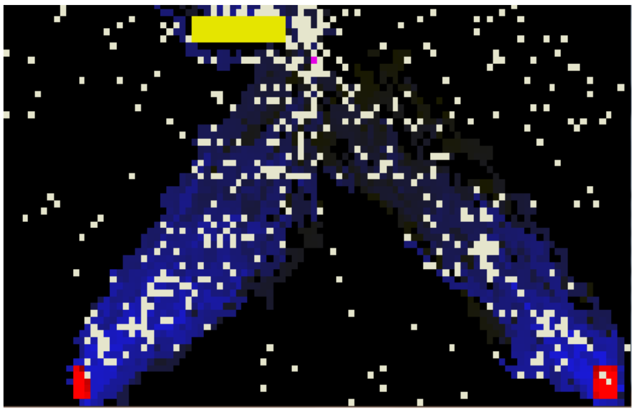
Figure 1. Ant Colony with two

The ant colony simulation was then evaluated by comparison to results in literature.

6 EVALUATION

To evaluate the developed system the best possible way was to firstly run tests using the algorithm implemented and compare the values and outputs to competitors using ACO or other algorithms to solve RPP. It was also worth comparing how the programming language or changes to the metaheuristics are used as these could possible effect performance. A good starting comparison was to compare the results achieved to work presented by (Dorigo, Maniezzo, & Colorni, Ant System: Optimization by a Colony of Cooperating Agents, 1996) in his paper or results found by (Perumal, et al., 2016) and then furthermore to other more recent variations.

7 CONCLUSIONS & FUTURE WORK

In conclusion ACO shows many potential solutions to solve the problem of RPP. Although ACO does show great ability in solving optimization problems such as TSP and RPP it has shortcomings such as falling into local optimum or not converging quick enough. Many improvements can be made not only to the heuristics itself, but also by creating a hybrid solution using a combination with other heuristics.

A further area of study within the field however would be to consider the problem of multi-objective ACO. As a problem that could occur is if two food sources or end points exist, not only would the shortest path to each must be found, but then these would have to be compared to find which is the optimal solution.

Future work could include extending this to other platforms such as embedded systems. Furthermore looking into applying the algorithm with other languages such as C++ would be ideal as this would provide much greater time efficiency due to its compiled nature and also being a low level language.

Finally another area of interest would be the implementation of evolutionary aspects of GA systems so that it further optimises route planning.

REFERENCES

[1] Cao, J. (2016). Robot Global Path Planning Based on an Improved Ant Colony Alogrithm. Journal of Computer and Communications, 4, 11-19. doi:http://dx.doi.org/10.4236/jcc.2016.42002

[2] Châari, I., Koubâa, A., Bennaceur, H., Trigui, S., & Al-Shalfan, K. (2012). smartPath: A hybrid ACO-GA algorithm for robot path planning. Evolutionary Computation (CEC), 2012 IEEE Congress on (pp. 1-8). Brisbane, QLD, Australia: IEEE. doi:10.1109/CEC.2012.6256142

[3] Collins, R. J., & Jefferson, D. R. (1990). AntFarm: Towards Simulated Evolution. Los Angeles: University Of California. Retrieved from https://www.researchgate.net/profile/David_Jefferson/publication/2551611_AntFarm_Towards_Simulated_Evolution/links/0fcfd50928f91e0fda000000.pdf

[4] Dehuri, S., Ghosh, S., & Cho, S.-B. (2011). Integration of swarm intelligence and artificial neural network. World Scientific.

[5] Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (pp. 1470-1477). Washington: IEEE. doi:10.1109/CEC.1999.782657

[6] Dorigo, M., Birattari, M., & Stutzle, T. (2006, November). Ant colony optimization. IEEE Computational Intelligence Magazine, I(4), 28-39. doi:10.1109/MCI.2006.329691

[7] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26(1), 29-41. doi:10.1109/3477.484436

[8] Perumal, N., Rashid, R., Elamvazuthi, I., Tageldeen, M. K., Khan, M. K., & Parasuraman, S. (2016). Mobile robot path planning using Ant Colony Optimization. 2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA) (pp. 1-6). Ipoh: IEEE. doi:10.1109/ROMA.2016.7847836

[9] Roach, C., & Menezes, R. (2008). Handling Dynamic Networks Using Evolution in Ant-Colony Optimization. New Frontiers in Applied Artificial Intelligence (pp. 795-804). Wroclaw: Springer. doi:https://doi.org/10.1007/978-3-540-69052-8_83

[10] Yu, L., Wei, Z., Wang, H., Ding, Y., & Wang, Z. (2017). Path planning for mobile robot based on fast convergence ant colony algorithm. 2017 IEEE International Conference on Mechatronics and Automation (ICMA) (pp. 1493-1497). Takamatsu: IEEE. doi:10.1109/ICMA.2017.8016037