Independently Optimised Artificial Neural Network Interactive Searching Algorithm (IOANNIS)

Thomas Lawson, *up780962*, and Yordanka Popova, *up782716*.

(Coursework for ENG621 - Artificial Intelligence)

***Abstract*—Over the past few decades the density of compute power has dramatically increased, whilst at the same time it's cost has dramatically decreased. As a result, much research has been done into the field of artificial intelligence, allowing it to be taken from a concept to a working product thanks to the increased capability and availability of compute power.**

This research will look to propose and analyse an approach to interactive and real time route planning for a robot designed for a capture the flag like task, meaning it will have to locate and retrieve an object returning it to a designated point, whilst all the time avoiding people moving in it's path.

This paper presents and analyses an approach to developing a real time navigational algorithm, named IOANNIS. The paper will present and critically analyse the algorithm's ability to effectively train an artificial neural network using a genetic algorithm, and the time it would take to come up with a viable neural network capable of safely and effectively completing the task.

***Index Terms*—Artificial Intelligence, Path Finding, Artificial Neural Networks, Genetic Optimisation.**

I. Introduction

WITH this paper we hope to introduce an approach to effectively train a tracked robot to successfully navigate a foreign environment in order to retrieve and return an object.

Currently for many tasks carried out by robots a human is required to control and monitor it's operation. This can be expensive and time consuming. However the human touch can be seen as necessary as humans have a better inherent ability to react to unplanned events in the environment.

Autonomous robotic navigation is not a novel concept however, and in recent years with the reduction in price of computing and increase in density of computing power many companies have successfully demonstrated autonomously navigating systems capable of reacting to a real-world environment. For example, Boston Dynamics have created their handle robot, capable of box handling in a warehouse [1]. Or there's Tesla, who have created an autonomous self driving car [2].

The objective of this research is to present and evaluate an approach for training an Artificial Neural Network quickly and effectively on any platform it is presented with, so long as it's interactions are well defined. We hope this research will help tackle an issue with Neural Networks where they will be able to handle the environment they are trained in, and not work well beyond it. With faster training times

This paper proposes our algorithm IOANNIS and hopes to demonstrate an adaptable approach to the development of a navigation system powered by Artificial Intelligence. It will be meant to be portable, meaning it can be trained on any device, and do so in a fast and efficient manner. The navigation system it produces should be able to avoid changing and moving obstacles in a foreign environment, while searching for an object it is meant to retrieve.

For the purpose of this research a simulation will be created using Unity, with the simulated robot being based on the IRIS robot (figure 1) provided by the University of Portsmouth.

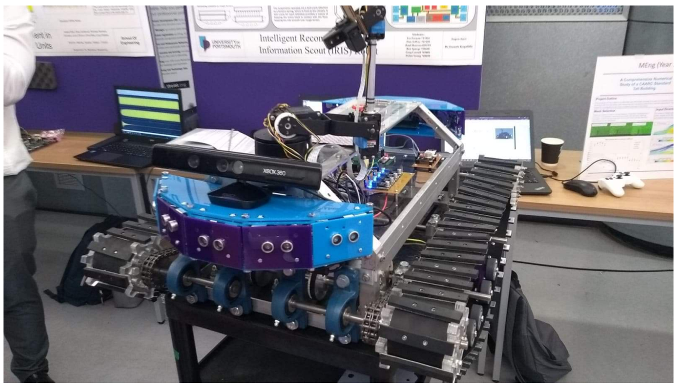


Fig. 1. Iris Robot

The rest of this paper will be organised as follows. Related work will be critically reviewed in section II. Section III will discuss the design of the algorithm IOANNIS. Section IV will discuss the implementation of the algorithm. Experimental results will be presented in section V. Lastly, section VI will present our findings and conclusions, as well as propose areas for future work.

II. Literature Review

Research has been done into multiple ways that robots can navigate [3]. Two major ways in which one can categorise navigation is by map-based and mapless navigation. In mapless navigation no prior knowledge of the environment here, much like the IRIS robot there is no prior knowledge of the environment and the robot is expected to navigate and explore based on reactions to sensory inputs.

Reflex agents have also been used as an approach to robotic navigation [4]. In this particular approach the robots direction of travel is determined by virtual attractive and repulsive forces created by the robot's knowledge of the environment created using it's sensors. This is an attractive approach and can be somewhat applied to this project's task as it can take advantage of the sensors to influence the robot's behaviour. However, using a reflex agent requires the ability to define the problem. This is beneficial because you can define how the robot should react, however the downside is the need for a clear understanding of the problem. Other approaches that involve the robot learning are are exciting as they can learn to deal with situations that may not have been envisaged.

Alternatively, if the environment is known there are techniques such as those used in games that could be applied. A relatively well known method used in computer games is the A* algorithm [5]. The A* algorithm is a searching algorithm similar to Dijkstra's algorithm. The difference is the algorithm has some reference to where it is ultimately going to guide the search in more complex spaces. While this has the drawback of only necessarily working in environments that are mapped out, this can lead to a more direct route than an algorithm that has to explore first.

Further improvements over A* are to restrict the search space such that it will not explore areas that will not lead to a solution [6]. Research has proposed heuristics that can rule out areas that will not lead towards the goal. In the task modelled here that could mean accessing other rooms who do not contain the goal, and do not lead anywhere else. Being able to rule these out in advance would significantly reduce the search space.

Artificial neural networks are an exciting prospect for navigation problems [7]. Opposed to other methods a neural network does not require previous definition of the problem. Instead, you must define a series of inputs and outputs. Then the algorithm should be trained such that the outputs it gives align with the trainer's desires. In a neural network a neuron will multiply an input value by it's weight to create an output. A series of these neurons are connected together to form a the neural network. This would be a beneficial approach as the system can be improved based on it successfully performing the task, therefore complex behaviour could be modelled without the ability to fully define the problem. However, this has a drawback in that it will take time to train, and it is not necessarily possible to predict how it will react, this could be dangerous if used on a robot around members of the public.

Genetic algorithms are an approach of solving a problem in the form of a fitness function my minimizing it's value. This has been applied to neural networks used for pathfinding in drones as if one can define the characteristics they are looking for in the resulting route in the form of a fitness function then the neural networks weights can be honed using a GA [8]. This could prove beneficial, as if a neural network were selected a method of training it's weights would be required. The fitness function could score the robots performance, and the chromosome could be the NNs weights.

Again, further research has been done to look into training a neural network for use guiding a robot [9]. This example looks at how practical examples can be used, not just training in a simulation. This has shown that it is possible to use this approach to train a neural network for robot navigation. It again however highlights the issue with this approach. The issues being that because of the nature of the algorithm it is impossible to predict the exact behaviour of the robot when it is presented with different obstacles.

Research has also looked at how a solely a genetic algorithm can be used to produce a route for the navigation of drones [10], [11]. This approach defines the route as a sequence of points before evaluating how fit they are according to length and avoidance of hostile radar amongst other values. This approach is valid and introduces interesting metrics that could enhance our fitness function. Although this approach claims to be capable of obstacle avoidance, the risk with more complex routes is that this would not be able to find an appropriate new route in time whilst it carries out it's evolutions. The benefit of using a GA to train an NN is that the Neural Network can carry out the decision making in near real time, the GA is only used during the training phase.

This algorithm will take advantage of a GA to train a NN. This method has been selected as it gives the best malleability, as once the objectives are laid out the robot simply has to be run and ranked according to how well it meets those objectives.

III. Design

A. Objectives

The main purpose of this research is to develop an approach to quickly training a robot to navigate successfully. To this end we will define the objectives of our research as follows.

1) The algorithm should be capable of training a neural network to navigate an autonomous robot.
2) The algorithm will be able to successfully train the robot in a quick and efficient manner.
    a) Where successfully train means to to be able to accomplish the retrieval task in a safe manner, meaning it does not collide with anything in the environment. Also meaning as to be able to accomplish it's task in an expedient manner. i.e. complete its task without unnecessary movements or stopping.
    b) And quick and efficient meaning a successful candidate neural network being produced within an acceptable number of generations.
3) The algorithm should be portable. Meaning the algorithm should not just be able to train for one type of robot. For example, the algorithm should be able to train on robots with reversible tracks, four wheels, etc.

B. IOANNIS Algorithm structure

For training the neural network, a genetic algorithm was selected. This is because with a task such as navigation, it is hard to use back propagation for training, as that would require for each decision, the actual correct decision to be determined and then re-train the network. When using a genetic algorithm to train a fitness function can be defined, this can evaluate the performance of the algorithm, and then whichever weights work better towards the defined goals will be kept on to further improve the robot.

The stages of the algorithm can be seen in figure 2 and can be summarised as follows.

1) Produce an initial population of weights.
2) For each set of weights in the population initialise a neural network using those weights and use it to operate the simulated robot.
3) Score the simulated robots and provide feedback to the genetic algorithm.
4) Using feedback from the simulation the genetic algorithm will use its fitness function to rank each set of weights.
5) If the end condition is met, return the fittest set of weights.
6) Otherwise crossover or mutate the weights according to the crossover and mutations rates.
7) Return to item 2 and repeat the process with the new population

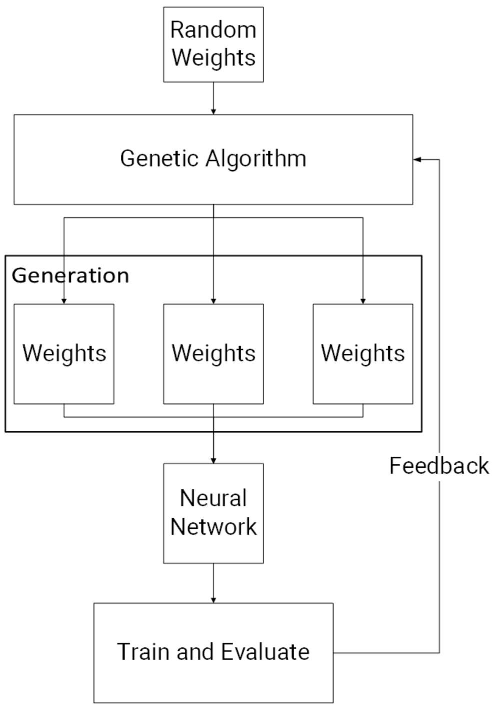


Fig. 2. Stages of the IOANNIS algorithm

The following sections will outline the design of the neural network and genetic algorithm that will train it.

C. Neural Network Design

The neural network will use l layers an input layer I and output layer O and x hidden layers H_x. The weights for each node W_{li}, where l represents the layer and i represents the weight's index, will be populated and trained using a genetic algorithm described in section III-D below.

There will be two output nodes. Each output will have applied to it a tan-sigmoid transfer function (figure 4) to smooth the output. We have selected the tan-sigmoid transfer

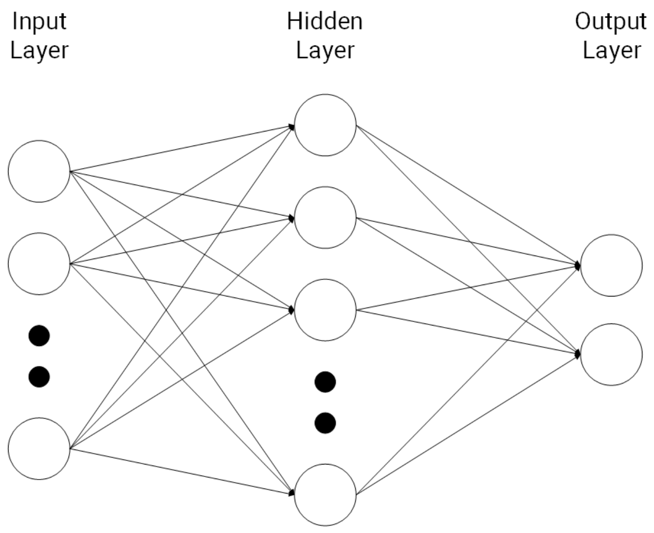


Fig. 3. Representative Neural Network

function as one output will represent each track, with a positive output directing the track to run forward, and a negative output directing the track to run backwards. The magnitude of each output will be used to control the speed at which the track runs.

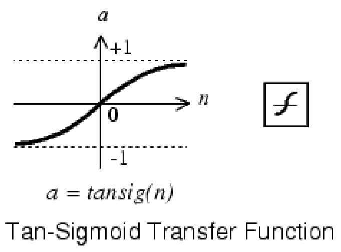


Fig. 4. Tan-Sigmoid Transfer Function

D. Genetic Algorithm Design

The genetic algorithm will be used to hone the weights used by the neural network described above.

1) Genetic Operators: Due to the nature of this task, we have elected to use three genetic operators. We have selected the three operators below, as due to the nature of this task, there is no need to use the operators for coercion where there are restrictions placed on an individuals viability. All candidates produces will be viable.

1) *Crossover Operator:* This will select another individual from the population and split it at a randomly selected point. The first section of genome A will be combined with the second section of genome B, and the first section of genome B will be combined with the second section of genome A. This will produce two children to be passed to the next generation.
2) *Mutation Operator:* This will randomly select a weight in the individual's genome, and randomly assign it a new

weight between 0.00 and 1.00. The resulting child will then be passed into the next generation

3) *Pass-through Operator:* This operator will pass the selected individual directly through to the next generation without change. This is the default operator and will be selected if the other two are not.

Each operator will have a chance of being selected proportional to it's selection probability.

2) Fitness Function: The fitness function to evaluate the fitness of an individual $f(I)$ shall be defined as follows.

$$f(I) = c + t \tag{1}$$

Where c represents the number of collisions the robot makes, and t represents the time it takes to complete it's task.

3) End Condition: To determine when the optimal solution has been found, there are two options. Firstly, it is possible to limit to a fixed number of generations, after which the fittest individual will be presented as the optimal solution. The other is to define an end condition, for this algorithm that could be defined as having an adaptable number of generations with at least one genome resulting in no collisions.

For the purpose this algorithm has been produced for, it is pertinent that the selected outcome prioritises safety of the robot and it's surroundings, as such the proposed algorithm will used a goal based end condition, and not be limited by the number of generations. This is because if a genome leads to collisions, this would be undesirable and unfit for purpose.

IV. Implementation

We will now discuss the implementation of our algorithm. Here we will discuss how the robot will be simulated in software, and we will also discuss how the robot will use a genetic algorithm to train a neural network to navigate.

A. Training

The robot is trained using a genetic algorithm. This genetic algorithm will create a population of chromosomes, these will simply be a sequence of doubles representing the weights for our NN. The algorithm will then initialise a NN with each set of weights. Once created these NNs will be allowed to run through the scenario controlling the robot. Once the generation has completed all it's runs, the chromosomes will be ranked according to their performance in the scenario. After ranking the robots will be selected using a roulette wheel selection method. This will mean that stronger characteristics should be carried forward, however it may also pick up from weaker performing chromosomes too, which should preserve some diversity to prevent stagnation at a local minimum. The selected children will then be either passed through back to the next generation, or they will be mutated and crossed over, according to the mutation chance. These are then joined with the population where the weakest candidates are then removed. The process is then repeated until the end condition is met. Which is either a suitably fit candidate, e.g. does not collide; or a suitable number of generations has passed.

Direction	Output
Left	0-2.5
Forward	2.5-5
Right	5-7.5
Backward	7.5-10

TABLE I
Output mapping to movement.

B. Robot Control

The robot will be able to move around a grid and able to choose to move forwards backwards left and right. It will also have visibility in these squares too.

These decisions as to where to move in the grid are formed by a neural network. The neural network has 4 inputs, which represent the content of the squares directly in front and behind, and to the left and right. From this the NN will create a single output of a value between 1 and 10. The output will then be used to determine which way to move as described in table I.

C. Simulator

The simulation software is written in C#. The robot will be simulated in a two dimensional grid shown in figure 5.

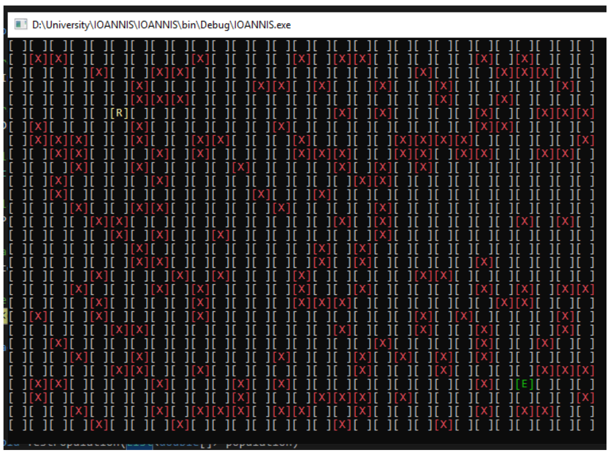


Fig. 5. Training UI.

In this grid the 'X' symbol represents obstacles to be avoided, 'R' represents the robot, and 'E' represents the goal.

V. Results

Firstly in figure 6 we can see the progress a robot (R) has made towards the target (E).

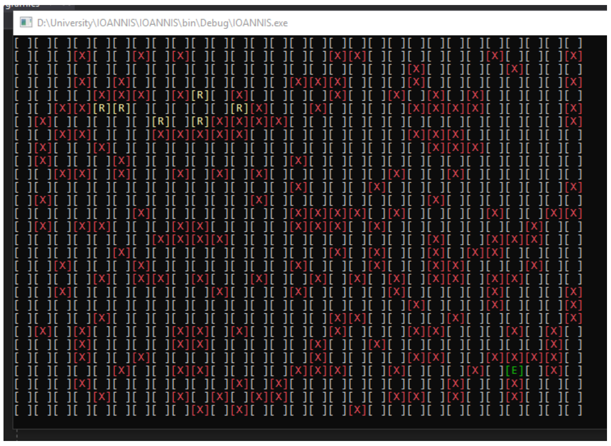
The Rs clustered in the upper right hand corner represent each of the robots trying to find their way to the objective.

We can then see more progress having been made in figure 7. This demonstrates how much improvement can be made over a short number of generations.

While initially the behaviour may seem random, over generations will start to follow a similar trend.

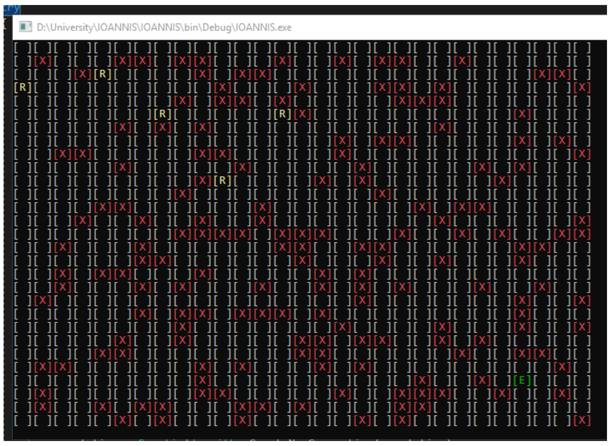
Then in figure 9 we can see that progress can continue even further.

Lastly **??** Shows a robot having successfully captured the flag at the bottom of the image at generation 21.



D:\University\IOANNIS\IOANNIS\bin\Debug\IOANNIS.exe

Fig. 6. Initial robot progress.



D:\University\IOANNIS\IOANNIS\bin\Debug\IOANNIS.exe

Fig. 7. Robot progress at generation 4.

A. Evaluation

This project has shown that with enough time a robot is able to complete it's objective in relatively few generations - 21. This has given us useful knowledge to be applied to future research. This has also shown that with even more attention to the fitness function, even better results can be achieved. However, that said, as can be seen by figure 9. With the correct ammount of time, this algorithm definitely shows improvement.

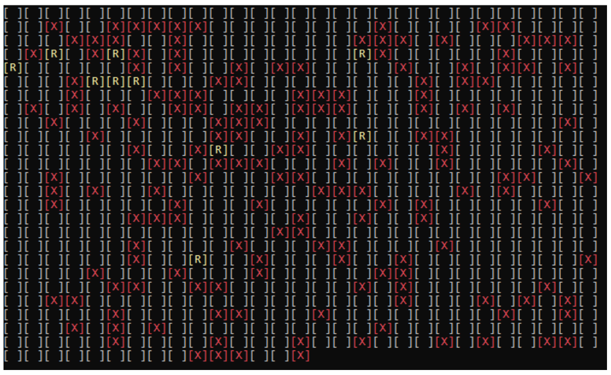


Fig. 8. Continued evolution.

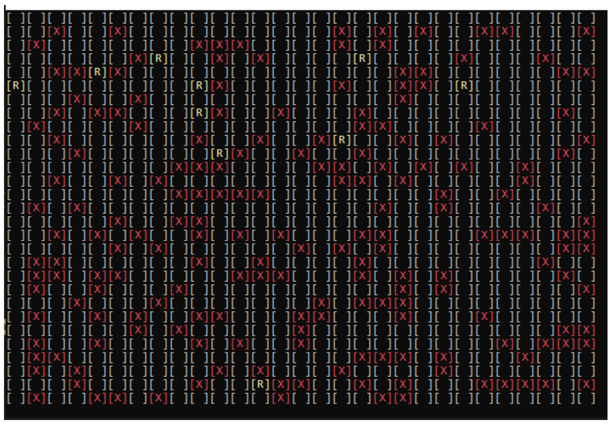


Fig. 9. Generation 21 flag capture.

VI. Conclusions

This paper takes a look at approaches to finding a route through an environment, and effective ways to train a robot to navigate.

The objective of this paper was to find a way to effectively teach a robot to navigate in a foreign environment. This paper has taken steps in the direction of discussing methods for autonomous navigation and discussed the pros and cons of each. It is safe to say that depending on whether a robot is mapless or not can have an impact on the approach taken. For our example where the robot was only required to navigate Anglsea building, the approach of making a reactionary robot without knowledge of the building was unnecessary. A potentially smarter approach would have been to create a robot who had knowledge of the environment and that would only react when presented with an obstacle it did not know about to path around it.

However, with that said this research did successfully show improvement through a number of generations for a robot autonomously navigating and avoiding obstacles in an environment of which it had no prior knowledge.

Future research should look at ways to further combine these efforts. Potentially creating robots that will be able to navigate maplessly, however will also be able to build a map as they navigate and use this to aid future trips to the same area.

A. Contributions

Both students contributed equally to this project.

References

[1] Boston Dynamics, "Handle — Boston Dynamics." [Online]. Available: https://www.bostondynamics.com/handle

[2] Tesla, "Autopilot — Tesla UK." [Online]. Available: https://www.tesla.com/en_GB/autopilot

[3] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual Navigation for Mobile Robots: A Survey," *Journal of Intelligent and Robotic Systems*, vol. 53, no. 3, pp. 263–296, nov 2008. [Online]. Available: http://link.springer.com/10.1007/s10846-008-9235-4

[4] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, sep 1989. [Online]. Available: http://ieeexplore.ieee.org/document/44033/

[5] X. Cui and H. Shi, "A*-based Pathfinding in Modern Computer Games," Tech. Rep. 1, 2011. [Online]. Available: https://www.researchgate.net/publication/267809499

[6] Y. Björnsson and K. Halldórsson, "Improved Heuristics for Optimal Pathfinding on Game Maps," Tech. Rep. [Online]. Available: www.aaai.org

[7] A. Deb, "Introduction to soft computing techniques: artificial neural networks, fuzzy logic and genetic algorithms," *Soft Computing in Textile Engineering*, pp. 3–24, jan 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781845696634500012

[8] S. A. Gautam and N. Verma, "Path planning for unmanned aerial vehicle based on genetic algorithm & artificial neural network in 3D," in *2014 International Conference on Data Mining and Intelligent Computing (ICDMIC)*. IEEE, sep 2014, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/6954257/

[9] D. . Floreano, F. Mondada, and D. Floreano, "Automatic Creation of an Autonomous Agent Genetic Evolution of a Neural Network Driven Robot Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot." [Online]. Available: https://doi.org/10.3929/ethz-a-010111549

[10] C. Zheng, L. Li, F. Xu, F. Sun, and M. Ding, "Evolutionary route planner for unmanned air vehicles," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 609–620, 2005.

[11] X. Zhang and H. Duan, "An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning," *Applied Soft Computing Journal*, vol. 26, pp. 270–284, jan 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494614004992