Artificial Intelligence – Report s5103250

Task 1

Task 1 is code-based and can be found in the *'Astar' Visual Studio Project*.

Task 2

Pathfinding in video games can use a multitude of approaches in order for an agent to traverse from one section of the play space to another. A good pathfinding algorithm should be able to efficiently calculate the shortest or least costly path between these two points without being too draining on the CPU. A* is very commonly used due to these factors, but also because it can be used in wider scenarios. There are three key types of learning strategies in machine learning: Supervised, Unsupervised and Reinforced Learning. This segment of research will evaluate three methods of pathfinding algorithm: Artificial Neural Networks (ANNs), Reinforcement Learning (RL), and Genetic Algorithms (GA), all of which use a different learning approach.

An Artificial Neural Network is a data processing method in which the code is structured to act like the human nervous system. They are a collection of singular neurons (perceptron) that are interconnected and layered that can be trained to recognised patterns within inputs [a1]. One of the most common cases of this is their use within medical examinations; they can detect potential anomalies within the body. ANNs use a learning rule called back-propagation, which circulates error back to the neurons to adjust their respective weights. Once trained, ANNs have the ability to deal with non-linear data, its speciality is with large number inputs [a2]. This means that ANNs themselves are not in fact too useful in pathfinding but are instead more efficient when it comes to classification problems. On top of this, ANNs are very computationally expensive $O(n^5)$ [a3]

RL is an unsupervised learning algorithm in which an agent learns just as a human would; through environmental interactions, i.e. learning by making and correcting mistakes. This means that they learn implicitly and can be used to solve very complex problems that cannot be solved through the use of conventional techniques [r1]. Most commonly, RL is applied to robotics, where an agent can be trained to perform a specific task. [r2] shows Reinforcement Learning being used to train a simulated car how to drive alongside multiple other examples of the applications of RL. Reaching an end goal is propagated backwards to mark the decisions made across that path, with each state being evaluated as either a reward or punishment. For a fixed start and end point, such as the ones provided in Task 1, Reinforcement learning has an advantage as it can exploit the data it has accumulated thus far. In terms of exploration, RL explicitly allows the probability of choosing new paths and actions.

Genetic Algorithms are a sub-strand of evolutionary algorithm based on Darwin's Theory of the same name. GA aims to find the optimal (or close to optimal) solutions to a problem. The objective (in this case, pathfinding) is discovered using 'mutations' in the agent that occurs every generation. For example, a string of binary digits can be used to direct the agent, (where 01 = left, 10 = right, 11 = up and 00 = down) and this can be used to evaluate its 'fitness'; how close it gets to the end goal [g1]. The next generation of agents will be the offspring of two of the previous agents and will replace the weaker individuals. This can be a very costly process and may take a long time until optimum offspring are produced that can heuristically reach the desired destination. In society today, GA has use almost everywhere, from Finance and Economics to music. "Genetic Algorithms (GA) seems to be a suitable approach for generating musical compositions. Combination of genetic operators (mutation, selection and crossover) in some ways simulates the innovative process (as real

composing is), enabling continuous 'improvement' of the obtained results [(A GENETIC ALGORITHM FOR COMPOSING MUSIC, 2021)[G2]].

Out of the three techniques, I believe using Genetic Algorithms will be the best option. Although costly, through each generation agents get closer to finding the optimal solution.

Task 3

Task 3 is code-based and can be found in the *'GA Pathfinding' Visual Studio Project*.

Task 4

Although two separate programs can process the same input and release the same output, the methods in which they process the information is vastly different. Take GA and A-Star for example. My programs use the same text files and produce the same results, but their codes are not the same; so, which is the better code? Efficiency is the most vital aspect when it comes to comparing two programs. It is defined by reliability, speed and the coding methodology to give one the verdict over the other. To compare the time of both algorithms, I will need an in-engine method. The 'Stopwatch' property held within the "System.Diagnostics" namespace [t1] is perfect. This keeps an accurate time of how long it takes to find the finish. Using a stopwatch in real-time is incredibly inaccurate and would not be good practise. We can compare CPU usage from within the "diagnostics" window during runtime.

Before testing, I believe that A-Star will remain the greater pathfinding algorithm. Its use of cost to reliably calculate its path means that it quickly and reliably reaches the end with the most efficient route. Genetic Algorithm is incredibly strong, but the nature of breeding the chromosomes over a multitude of generations means that although it will eventually find the route, this process is very costly, both timewise and process-wise.

Task 5

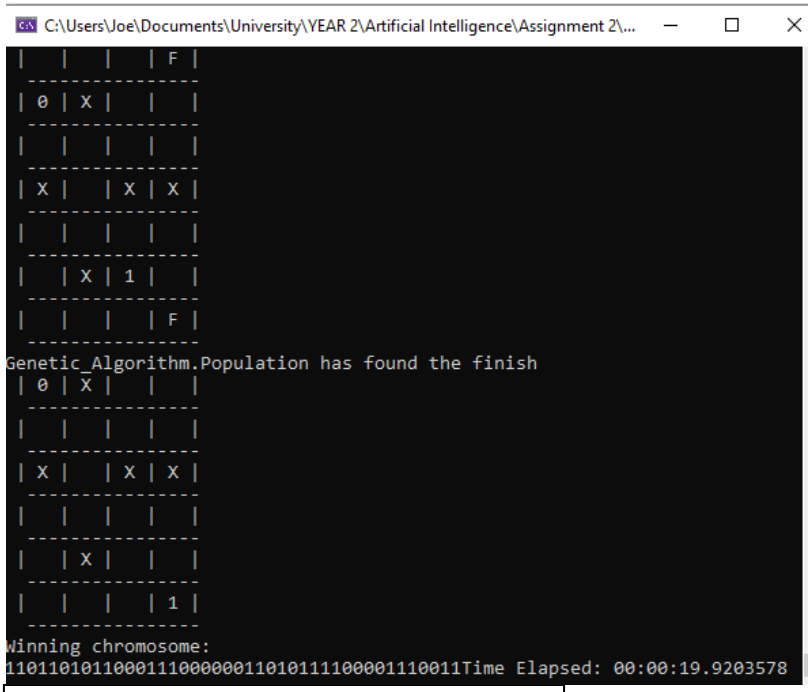

Figure 1: Time Elapsed Genetic Algorithm)

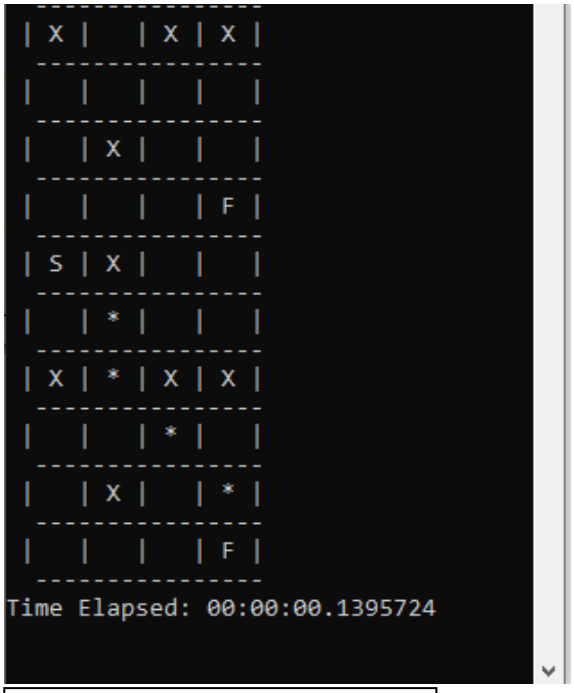

Figure 2: Time Elapsed (A-Star)

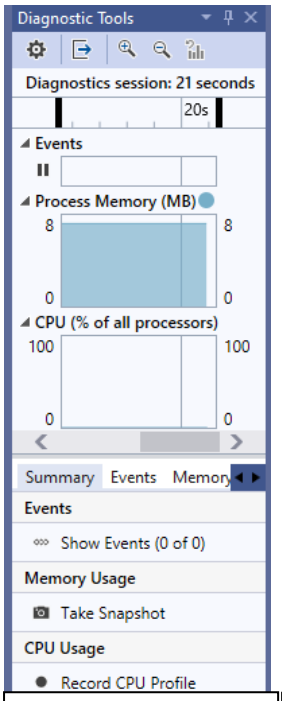

Fig.3 A-Star CPU usage

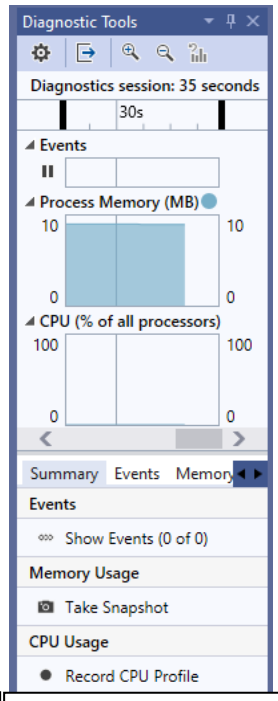

Fig.4 Genetic Algorithm CPU usage

As shown in Figure 1 and 2, the A-Star Algorithm finishes its processes in 0.139s, whilst GA took 19.9s. This proves outright that A-Star is the superior algorithm when it comes to speed. On the CPU usage side, A-Star uses 8MB of Process Memory at the end of the process (see fig.3) and GA uses 10 (fig.4).

In conclusion, A-Star is the greater pathfinding Algorithm. By calculating the cost of moving from one node to another, A-Star can make decisions that lead it towards the end goal without the use of iterative generations that “guesses” where it should travel. This means that only one agent is required to solve the maze, in comparison to the 4 agents in GA that are constantly testing their path and being crossed over with one another.

References

[a1] Investopedia. 2021. *Artificial Neural Network (ANN)*. [online] Available at: <https://www.investopedia.com/terms/a/artificial-neural-networks-ann.asp> [Accessed 12 January 2021].

[a2] Subscription.packtpub.com. 2021. *Pros And Cons Of Neural Networks*. [online] Available at: <https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788397872/1/ch01lvl1sec27/pros-and-cons-of-neural-networks> [Accessed 9 January 2021].

[a3] Fredenslund, K., 2021. *Computational Complexity Of Neural Networks*. [online] Kasperfred. Available at: <https://kasperfred.com/series/introduction-to-neural-networks/computational-complexity-of-neural-networks> [Accessed 9 January 2021].

[r1] Pythonista Planet. 2021. *Pros And Cons Of Reinforcement Learning | Pythonista Planet*. [online] Available at: <https://www.pythonistaplanet.com/pros-and-cons-of-reinforcement-learning/> [Accessed 9 January 2021].

[r2] neptune.ai. 2021. *10 Real-Life Applications Of Reinforcement Learning - Neptune.Ai*. [online] Available at: <https://neptune.ai/blog/reinforcement-learning-applications> [Accessed 12 January 2021].

[g1] Medium. 2021. *Introduction To Genetic Algorithms — Including Example Code*. [online] Available at: <https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3> [Accessed 12 January 2021].

[g2] 2021. *A GENETIC ALGORITHM FOR COMPOSING MUSIC*. [ebook] Banka Luka: Dragan MATIĆ, p.158. Available at: <http://elib.mi.sanu.ac.rs/files/journals/yjor/39/yujorn39p157-177.pdf> [Accessed 12 January 2021].

[t1] Docs.microsoft.com. 2021. *Stopwatch.Elapsed Property (System.Diagnostics)*. [online] Available at: <https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.stopwatch.elapsed?view=net-5.0> [Accessed 13 January 2021].