Drone Swarms to Predict Forest Fire Propagation

Jumaira Miller 983101
October 2020

Initial Document submitted to Swansea University
In Partial Fulfilment for the Degree of Bachelor of Science
Department of Computer Science
Swansea University

1 Table of Contents

2 Introduction

F

Forest fires have a devastating effect ecologically, socio-economically, and atmospherically. For example, ecologists believe that Australia's 2020 bushfire season has driven many endangered species to extinction [1] [2]. Some of the local social-economic effects include casualty of civilians, blocking public transportation, cutting telecommunication networks, and ravaging homes and local industries [3]. Australia 2019-2020 bushfire season left more than 5000 homes destroyed across multiple fire zones. This same bushfire season is recorded as one of the country's costliest to date.

It is also common knowledge that forest fires have a clear detrimental effect on climate change. However, it is often overlooked that the reverse is also true. The rise in temperature results in drier soil and vegetation, which in turn results in more frequent fires, and prolonged forest fire seasons. The effects of this feedback loop exacerbate the issues with each cycle. More emissions are released from forest fires, leading to warmer temperatures, which result in more profound effects of climate change, the effects of which appear during wildfire season [4].

This vicious cycle means the frequency of forest fires will only escalate over time. However, the sudden rise this year was unprecedented. As of August 2020, the frequency of forest fires has increased by 13% compared to the entirety of 2019. Not only have we seen an increase in the frequency of the fires, but also in intensity. The Arctic Circle, Australia, and the state of California are just some of countries and states that have reported record fires this year.

There is a growing sense of urgency within the scientific community on how to deal with this problem. One area is modelling and then predicting the propagation of forest fires. This is what my project will cover.

2.1 Motivation

This practical urgency to address and mitigate the impact of forest fires is the first motivation for this project. The urgency demands fast and reliable solutions which we can get from data-driven analysis.

Machine Learning (ML) has long been used in Data Science for data analytics which aims to automate the extraction of key insights from large sets of data. Some common challenge with the development, testing, and training of ML models are concerned with the issues related to Big Data: increasing volumes, dynamic changes, and noise in the datasets. Traditional model-based methods limit the problems that may be solved using ML to those that can be decomposed into either continuous or differentiable functions. An alternative is to use data-driven approaches [5].

Swarm Intelligence (SI) is a type of meta-heuristic algorithm which require little to no prior assumptions. This means they are able to handle high-dimensional and dynamic problems for which it is either difficult or not possible to obtain derivatives, including discreet, non-continuous, or noisy functions.

Another point of motivation is that it is not always possible to obtain large datasets for novel problems, or for problems that require specific attributes. One of the major problems in ML is obtaining high-quality data. However, this problem can be very easily mitigated through the use of computer simulation. With the recent rise in advancements in computer vision, processor power, and computer parallelisation, game engines are able to simulate near real-world data [6]. Not only do simulations remove the need for high-quality data (with attributes specifically tailored to the given problem – which is not always possible), but they also allow for a more effective way of communication knowledge gained from ML models. As such, an aim of this project is to make use of an ML.NET, an ML framework that supports C#, to implement ML scripts to simulate, train, and visualise the propagation using Unity 3D game engine.

2.2 Project Outline

This project will make use of an off-the-shelf simulation script of a forest fire. Through this simulation, I will train an unsupervised ML model to obtain insights on the propagation of forest fires, to then be able to predict the propagation of an unseen forest fire simulation. The main reason for using a readily available Unity 3D simulation script is because it allows me to focus this project on Optimisation and ML algorithms. I will also be making use of SI for effective search algorithms and optimisation methods.

Listed below is a breakdown of the three identified stages of this project:

1) **Boundary Detection and Optimisation**
Using an aerial image of the state of a forest fire at a specific timestamp, I will first need to be able to detect the boundary of the fire. I will be treating this problem as a segmentation problem rather than an edge detection problem. The reason for this is because edge detection is sensitive to noise and will create a lot of unnecessary edges. Another reason for not using edge detection is because I will be using colour images as a result of how the simulation is created. Edge detection is typically not colour based, so an edge will be added. Segmentation algorithms would be a better alternative as they fundamentally work by trying to identify an approximate segment, so if a small portion of the fire is lost in the process, the algorithm is still able to capture most of the fire.
Once the segmentation algorithm is able to capture the boundary, I will attempt to optimise the algorithm using either spatial optimisation or particle swarm optimisation.

2) **Train an ML Model**
I will use the segmentation algorithm on several simulations to curate a dataset where each sample is a series of images at different timestamps of a single forest fire simulation. An ML model will be trained on this processed dataset of forest fires using an auto-encoder which makes use of two Neural Networks.

3) **Predict the Propagation of a Forest Fire**
The resulting weight from the auto-encoder will be fed into a predictor which should then be able to construct an image of the forest fire at a later timestamp.

It is important to note that the overall forest fire dataset will be split into a training, prediction, and testing datasets.

2.3 Aim

The aims of this project are as follows:

Primary Aims	- Modify an off-the-shelf forest fire simulation to be able to take in a dataset and continuously generate simulations (the script currently requires manual input of attributes to generate a single simulation). - Successfully implement an intelligent swarm system of drones with effective swarm behaviour for efficient boundary detection using a segmentation algorithm. - Learn the ML.NET library and be able to implement a basic auto-encoder, using neural networks, to train an ML model of forest fire propagation.
Secondary Aims	- Optimise the initial location-allocation of a swarm of drones for an optimal segmentation algorithm. - Implement a predictor using the resulting weight form the auto-encoder to be able to approximate the propagation of a forest fire after a given time.

I am conscious of the time constraints of this project, and thus will focus on achieving most of my primary aims first. Should this be successful, I will then attempt to make progress in implementing the secondary aims.

3 Background Research

3.1 Swarm Intelligence

The Maths and Sciences have long adopted algorithmic solutions and fields of studies from observing, modelling, and analysing nature and its natural phenomena. One such discipline, in the study of 'artificial life', was inspired by the collective behaviours of social insects and high order living animals. Examples of such animals include ants, a flock of birds, schools of fish, and bees. The collective behaviour of such animals is called '*swarm behaviour*' [7] [8] [9].

3.1.1 Overview

Swarm behaviour refers to the aggregation motion of a collection of decentralised, self-organising individual systems to accomplish a given task [8]. From modelling and analysis such behaviours, scientists have found that a collection of simple individuals, using simple local communication rules, can achieve goals which an individual may not have been capable of or achieve goals far more efficiently.

Swarm Intelligence (SI) is the application of the observed *robustness*, *scalability*, and *distributed self-organisation principles*, observed in natural swarm systems, to artificial systems [9].

Aligning with that of natural swarm systems, SI systems are characterised as follows [7]:

- Extensive collections of simple individual systems;
- Individuals must be near homogenous (not much, if any, variety in the type of individual systems);
- Individuals actions using a limited number of simple behavioural rules to interact, locally, with other individuals and the surrounding environment;
- Self-organisation of the overall SI system is achieved as a result of the collective actions of the individual systems

In SI systems, the collective group of individual systems is considered the primary system. Thus, the individual systems themselves are the subsystems. The central concept behind SI system (as a group of simple autonomous systems with limited capabilities) is that the collective knowledge of the system can obtain insights which can be used to solve complex problems (both continuous and discrete non-linear problems) which the simple subsets of the collective system would not have been able to solve themselves [10].

As a result of the characteristics specified above, SI systems have the following properties [9] [7]:

- Scalability — The systems can introduce more subsystems and maintain all original functionalities without the need to redefine the interaction rules between the subsystems. In other words, increasing the overall size of the system will improve performance. However, reducing the number of subsystems within the SI system will not affect the functionality of the system.

- Robustness — As a result of the scalability property and the way the subsystems interact, the system is tolerant to failure. The failure of a subsystem does not affect the functionality of the overall system; the failing subsystem can be removed or replaced with another one.

- Distributed Coordination — Each subsystem is restricted to local and simple sensing and communication abilities. The parallel actions (every subsystem simultaneously performing different actions at different locations) of the subsystems combined with the lack of a 'leading' subsystem or a central control system gives the SI system this property of having distributed Coordination

Evaluating the performance of SI systems have proven to be a challenge in the study of SI. Due to a lack of analytic studies this discipline, a general criterion on evaluating the performance SI systems is yet to be established. Instead, researchers and developers have taken to evaluate the performance based on other properties. An example of an alternative is assessing the flexibility of such systems, which is essentially a robustness property [8].

SI has been applied to a variety of different problems, including optimisation, communication networks, and robotics. Some prominent swarm-based algorithms include "Ant Colony Optimization Algorithms (ACO), Particle Swarm Optimization Algorithms (PSO), Artificial Fish Swarm Algorithm (AFSA)" [11].

3.1.2 Particle Swarm Optimisation

Particle swarm optimisation (PSO) is a stochastic optimisation technique explicitly inspired by the natural swarm behaviour of flocks of birds and schools of fish [11]. PSO aims to optimise a problem by iteratively trying to improve a candidate solution based on a given quality (decision variable). Each particle in a candidate solution is able to move around the search-space using the key properties **position** and **velocity** [12]. PSO is a metaheuristic algorithm [13] – it makes little to no prior assumptions about the problem being analysed. The general flowchart is highlighted below:

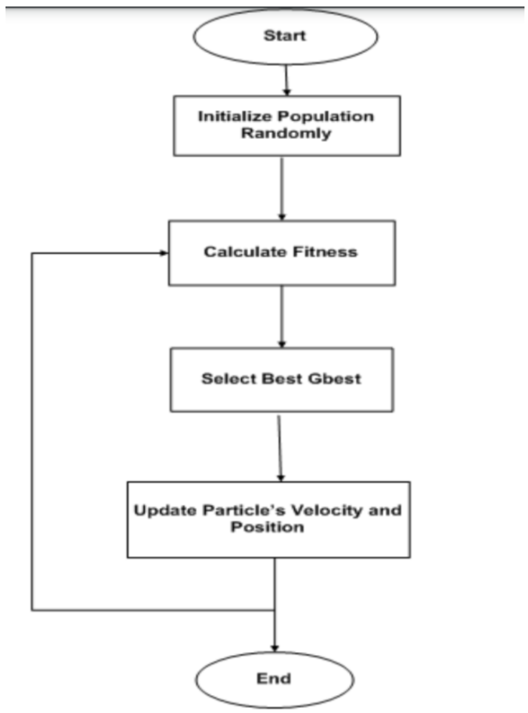

Figure 1: A flowchart Detailing the algorithm [13]

Before going to detail further, I will outline the key points to know about PSOs [13]:

- **Particle** — A particle can be defined as$P_i \in [a, b]$ where $i \ \in \{1,2,3,4,6, \dots, D\} \in \mathbb{R}$. D represents the dimension of the particle, and ℝ represents real numbers.
- **Fitness Function** — A function used to find the optimal solution. It is usually the objective function
- **Local Best** — Best position of the particle relative to all other positions visited
- **Global Best** — Best position achieve among all particles

In PSO algorithms, each solution is a particle which is able to move in a search landscape. To update the position of each particle, the following two vectors are required, and both are updated at each iteration:

- **Velocity Update** Vector determining speed and direction of the particle. The equation detailing velocity update is below. It is updated at each iteration:

$$\overrightarrow{V_i^{t+1}} = w\overrightarrow{V_i^t} + C_1 r_1 \left(\overrightarrow{P_i^t} - \overrightarrow{X_i^t}\right) + C_2 r_2 (\overrightarrow{G^t} - \overrightarrow{X_i^t})$$

= Inertia + Cognitive Component + Social Component

Where:

t	the iteration number
i	agent or solution ID
$\overrightarrow{V_i^t}$	current velocity
$\overrightarrow{X_i^t}$	position of current iteration
$\overrightarrow{P_i^t}$	personal best solution
$\overrightarrow{G^t}$	global best solution; does not need an *i* attribute because it concerns the whole system and not a particular solution/agent
$\overrightarrow{G^t} - \overrightarrow{X_i^t}$	distance from current position to global best
$\overrightarrow{P_i^t} - \overrightarrow{X_i^t}$	distance from current position to local best

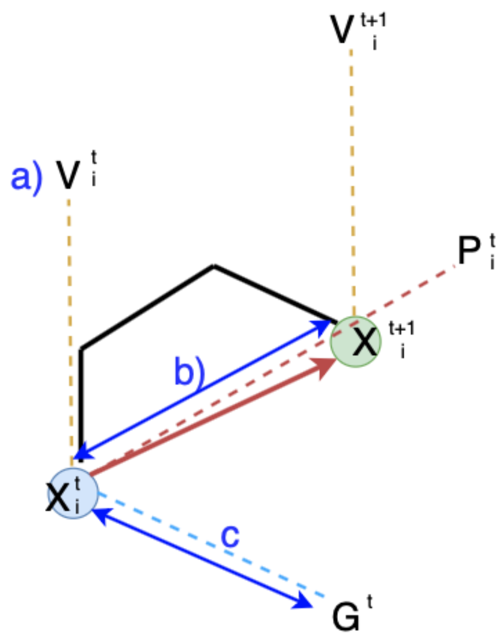

Figure 2: A diagram visualising each particle's iteration step

- **Position Update** Particles move toward the best position according to the fitness function. The equation detailing position update is below. The position of the next iteration is given by:

$$\overrightarrow{X_i^{t+1}} = X_i^t + V_i^{t+1}$$

The position of the next iteration is the sum of the agent's current position and next iteration's velocity.

For the PSO algorithm, variants are merely the changing of parameters. This can include starting initialisation, inertia weight (*w*) etc. [13]

PSO works by creating a swarm of particles that form a population. They are optimised using the objective function. Candidate solutions are recombined to produce a new generation of solutions with the best trait being inherited. This will continue until there are no further improvements or PSO is not able to find a solution. The goal is that over time the particles collectively move to the global best.

An example of inspiration for PSO is 'Flocking' [14]. This example was made to simulate the behaviour of birds. A bird-like object (boid) is defined by its position and a velocity vector. The idea is that with each timestep, each boid moves by a predetermined distance along the velocity vector. The boid will observe its immediate surroundings and depending on the result, will change its position and velocity vector. There are three key behaviours:

- **Separation**: All boids are expected to avoid collisions with each other and their environment. After an interval of time, all boids will check their personal spaces. Should any boid fall within this personal circle (smaller than the local peripheral grey space) the affected boid will move away.

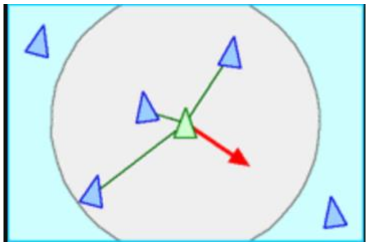

Figure 3: A visualisation of boid separation [15]

- **Alignment**: This behaviour rule ensures the boids steer towards the collective direction. At each timestep the boids will re-assess and readjust their alignment based on the average alignment of the boids within their vicinity.

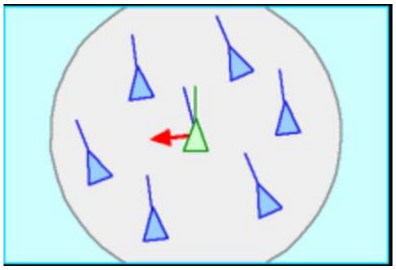

Figure 4: A visualisation of boid alignment [15]

- **Cohesion**: In this behaviour, the boids move as a group. All boids follow alongside their neighbours. At each timestep, a component, $\vartriangle V$, is added towards the centre of gravity for boids in the local vicinity.

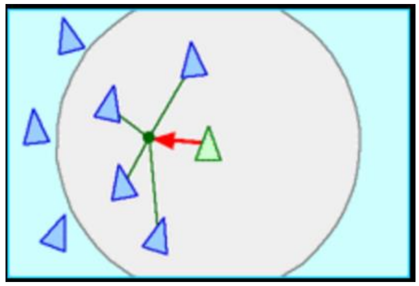

Figure 5: A visualisation of boid cohesion [15]

I intend to use this method of optimisation (PSO) within colour image segmentation to find the optimal solution.

3.2 Colour Image Segmentation

3.2.1 Overview

Partitional cluster analysis is the assignment of elements into subsets such that each subset is as homogeneous as possible, while the subsets between one another, should be as heterogeneous as possible [16]. Image segmentation follows this principle by identifying homogenous regions in the image, usually via the pixels themselves. Segmentation is useful in simplifying an image to make it easier to analyse. Typical use of image segmentation includes identifying boundaries. Methods for image segmentation were originally for greyscale image but can be extended to coloured images. However, more features must be considered such as colour space.

Notable examples of Image segmentation include K-Means which I will outline below:
K-Means clustering is a popular clustering technique. In this clustering technique there are *k* clusters. Each cluster is represented by a centroid. The centroids must be initialised at first but with every iteration, each data point is assigned to the centroid closest to them. The centroids are then updated so they are the mean value of the assigned cluster [16]. The aim is to minimise the squared error function which is shown below:

$$J = \sum_{c=1}^{k} \sum_{i=1}^{n^{(c)}} \left\| \vec{X}_i^{(c)} - \vec{C}_c \right\|^2$$

Figure 6: Square Error Function [16]

Where:

- $\vec{X}_i^{(c)}$ is the i^th data point assigned to cluster c.
- $\vec{C}_c$ is the centroid for cluster c.

The advantages of using K-means are that it is easy to implement and does guarantee convergence [17]. Whilst there are other forms of clustering such as Gaussian Mixture Model, due to the focus being on the optimisation of drone allocation, I believe using K-means is suitable. However, I am aware of the draw backs such as choosing the cluster number manually and will take this into consideration during implementation.

A possible solution to the problem of initial drone location allocation is using Particle Swarm Optimisation (PSO) in colour segmentation.

3.3 Spatial Optimisation

Spatial optimisation is the use of mathematical and computation techniques to find the best solutions to geographic decision problems using strictly defined parameters. It is a sub-field under the broader field of optimisation [18].

Optimisation problems consist of objective functions, decision variables and other constraints. Spatial optimisation focuses on the representation of space as an essential part in determining the decision variables and objective functions when solving the problem. The functions themselves will tend to represent physical relations such as shape, distance or whether two or more areas overlap. The general form of spatial optimisation is outlined below:

Given a set of geographic decision variables where $x \in X$ represents the location:

$$Optimise\big(f(x)\big)$$

With n constraints in the form:

$$g_n(x) \leq C_n$$
$$g_n(x) \geq C_n$$
$$x \geq d$$

Where C_n is the limit value for the nth condition.

In spatial optimisation, the decision variables represent geographical locations on a grid (x, y) and contain any number of dimensions concerning the problem at hand.

The challenge is the definition of these decision variables. Depending on the problem, it may require that the distance be spread in a uniform way to maximise coverage. Alternatively, clustering may be required. Each requirement comes with its complexity. Using clustering as an example, I would need to address issues of cells being adjacent or connected. This would, of course, require algorithms such as nearest neighbour [18].

For my project, it is likely I will be dealing with maximising the spread of the swarm-bots to allow them to uniformly spread out across the plane of the simulated forest fire. This optimisation method may be an alternative to PSO.

3.4 Related Works

Haghian's thesis on Deep Representation Learning for Forest Wildfires, details how she tackled the prediction of propagation on three phases [19]:

- Data Pre-processing: The dataset was run through a pre-processing algorithm to make the images more like satellite images; just the different colour variations were shown. The dataset was split into a combination of training, test and validation data
- Creation of a Deep Forest Wildfire Auto-Encoder (DFWAE): This made use of an artificial neural network, a convolutional Auto-Encoder (CAE) specifically, to learn how to understand the representation of the dataset; learning the overall structure of the forest fire. It is one extensive network made up of two sub neural networks. The encoder is a convolutional neural network (CNN), and the decoder can also be a CNN, or it can be a deconvolutional neural network (DNN). The model is below:

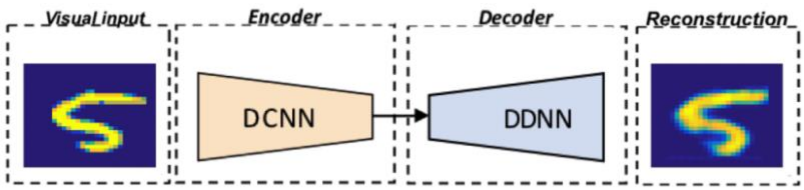

Figure 7: Model of the CAE [19]

- Deep Forest Wildfire Predictor (DFWP): This model also made use of a neural network to predict the spreading of the fire. Hagian used the weights to train the predictor, which was made up of the following:
    - A (Long Short-Term Memory) LSTM; a variant of a Recurrent Neural Network (RNN)
    - A CNN for feature extraction

Figure 8 outlines the above-described process:

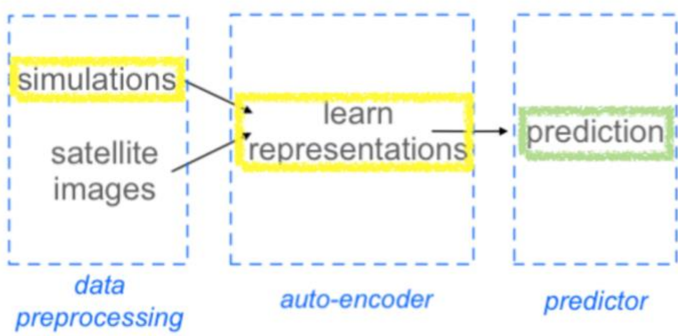

Figure 8: Schema on how to train the model on simulated data [19]

There are several takeaways from this project that will be of crucial use for mine:

- I will need to use or implement an auto-encoder, to learn the overall structure of the fire through the dataset containing the segmented data.
- I will also split the dataset into three sections: a training set, testing set and validation set
- The thesis did not implement the prediction phase. However, I will be aiming to implement this, and to do this I will need to make use of an RNN. RNNs are particularly useful for keeping 'memory' as they hold the previous state whilst producing the current state, which will then be used to predict the n+1 state.

4 Project Management

4.1 Proposed Frameworks and Technologies

Several frameworks and technologies will be required for the implementation and development of this project. Outlined below are some such tools and their usefulness to the project.

Visual Studio (VS)	A full-featured IDE developed by Microsoft, which supports 36 programming languages including C# and offers several features that enables efficient software development. One of these features is IntelliSense, which enables fast and accurate script development. Other helpful features include built-in Git support for version control and advanced debugging compatibility [20]. VS is also very compatible with Unity. By using the Implement MonoBehaviours and the Quick MonoBehaviours Wizards, I can directly curate Unity script methods through VS [21].
Unity 3D	Unity is a visual interface and cross-platform game engine primarily used in the development of video games. However, with the increase in processing capabilities of game engines, researchers have found that by applying ML techniques, they can visualise detailed simulations for their ML models. Unity has several features, such as the physics engine; enabling researchers to focus on the implementation of the model instead of spending time on the simulation setup. Another benefit to researchers using Unity to train ML models through simulations is that Unity also supports .NET, which is a framework I will need for this project.
Unity Machine Learning Agents (ML-Agents)	Unity has recently released an open-source plugin in beta, which allows the training of intelligent agents using several ML methods such as Reinforcement Learning, Neuroevolution, and Clustering, amongst many others [22]. One of the main benefits of ML-Agents in terms of training ML models is the ability to run multiple concurrent instances of a simulation on a single machine. This drastically increases the throughput of training samples and reduces the time required for training. However, the number of parallel environments we can run is limited by the resource constraints of the machine used for the simulation.
ML.NET	ML.NET is Microsoft's cross-platform, open-source ML framework. A feature of particular interest to me is NimbusML. It is a Python binding which allows developers to create and train ML.NET models in Python (enabling the use of libraries such as NumPy and Pandas, which I am already familiar with) which can be run natively in .NET [23].
Google Colab	Google Colab is a free cloud service which is compatible with Unity ML-Agent. Although Unity does also offer its own cloud service, Google Colab is a more familiar alternative to have. Regardless of which cloud service I use, training on a cloud service reduces the possibility of losing data and ensures computational power is not an obstacle.
GitHub	GitHub is a cloud-based repository service which makes use of Git's open-source version control system. I will be using GitHub to back up my work. It will act as a backup storage location and ensure the ability to 'roll back' to previous revisions when needed. It also allows me to incrementally 'push' code into the central repository by merging development branches. This will allow me to follow software engineering principles, and write sets of code that, after testing, can be integrated into the master branch.

4.2 Risk Analysis

Listed below, ranked by overall risk factor, are the relevant risks associated with this project.

4.2.1 Technical Risks

Risk	Effect	Mitigation	Severity	Probability
My machine may not be computationally strong enough to train the ML model	It is very likely that this may be the case as training is a very computationally demanding process. The impact is relatively high as it would halt progressing onto the next step of predicting the new boundary of the forest fire. The severity increases in proportion to the size of dataset and project files.	It is possible to train Unity ML-Agents on Google Colab. However, as Unity's ML-Agent plugin is still under beta testing, I may come across come issues. Another alternative is using Unity Simulation which is a managed cloud service with capabilities of funning thousands of simulations in parallel on the cloud.	High	High
Training the ML model for forest fire propagation may take longer than anticipated.	The reasoning behind why this risk is so severe is the same as the above. However, the probability of this risk is slightly lower as a long training time is already expected for ML modelling.	Unity ML-Agent provides a series of features specifically aimed to reduce training time in ML models. For the same reason as explained above, using Unity Simulation can help mitigate this risk. Reducing the dimensionality of the data such that no desired information is lost, can also reduce training time.	High	Medium
Loss of working code if further developments result in a bug that cannot be found or debugged.	It could potentially halt all further progress until the bug is dealt with. Alternatively, it may mean submitting or moving forward with code that does not work as it should. The probability is high as bugs are a very common occurrence in code development. The severity is medium as it will slow down progress until it can be resolved.	I will be pushing each working version to GitHub which is a widely used open-source version control system. I will also be following software engineering principles such as making use of break points and creating testable code. I will also be following the KISS principle (Keep It Simple Stupid) which will prevent introducing unnecessary	Medium	High

		complexity and thus reducing the likelihood of bugs.		
Loss of unsaved work	Project progress will be pushed back as work has to be reimplemented. This in turn will cause all other deliverables to be pushed further down the project timeline.	Visual Studio provide an extension, Auto Save File, to save changes as they are made.	Medium	Medium
Data corruption of files resulting in the loss of project progress	The severity is understandably high as it makes failure of the CSP-354 Project Planning and Specification module very plausible. At the very least, it would mean restarting the project and omitting some key deliverable. Although the severity of such an incident would be very high, the chances of it are low. This is because I regularly make regular backups.	Back data on multiple physical and cloud services. Specifically, I will be using GitHub repositories, Google Drive, Unity Cloud Build to monitor my source repository.	High	Low
Inability to optimise the location-allocation of the drone swarms if the complexity of the simulation makes the evaluation of the objective function	It may mean I will not be able to deliver fully on the aim of finding the optimal starting location of the drone swarms. The severity of this is low as this is only within the first stage of the project. It just means that I will move forward to modelling the propagation after finding the boundary of the fire, instead of optimising the segmentation problem. The likelihood of this is uncertain until an attempt has been made. Hence, why the probability is Medium.	If the problem cannot be optimised, there is nothing more that can be done to move forward in this particular aspect.	Low	Medium

4.2.2 Personal Risks

Risk	Effect	Mitigation	Severity	Probability
Inadequate experience working with programming language and Unity 3-D game engine	The effects of this risk are that I will be needing to spend more time covering tutorials, documentation, and familiarising myself with the relevant material. Even though I will be programming in C# for the first time, the severity of this obstacle is low as the language is very similar to one which I am familiar with. C# and Java are very similar.	I will need to allocate enough time to allocate sufficient time for the implementation of this project to account for additional time throughout the process to refer to documentations and other supporting material.	Low	Medium
Regulated internet access in Qatar	Due to personal reasons, I will be travelling to Qatar in December. Some content is unavailable to access in from within the country as the government strictly regulate internet access. This may affect progress in term of research.	This has been an issue within the country for some time. However, it can be easily mitigated by using a VPN to access such content	Low	Medium
I may get diagnosed with a severe illness (e.g. Coronavirus)	Although I may be ill for at most two weeks, the severity of this risks effect on this project is relatively low. Almost all work is remotely done and can be continued whilst in isolation for that period of time.	My flat and I are taking all necessary health and hygiene precautions. I have also significantly minimised face-to-face interactions.	Low	Low

4.3 Project Timeline

The Gantt chart visualised below in Figure 9 outlines a proposed timeline for this project. Each field represents a week, labelled by the week beginning day. The red lines are an indicator of when key deliverables are expected to be complete. Secondary aims have intentionally been excluded from the timeline (outlines in section 1.3) to be able to first ensure that enough time has been allocated to achieve the primary aims.

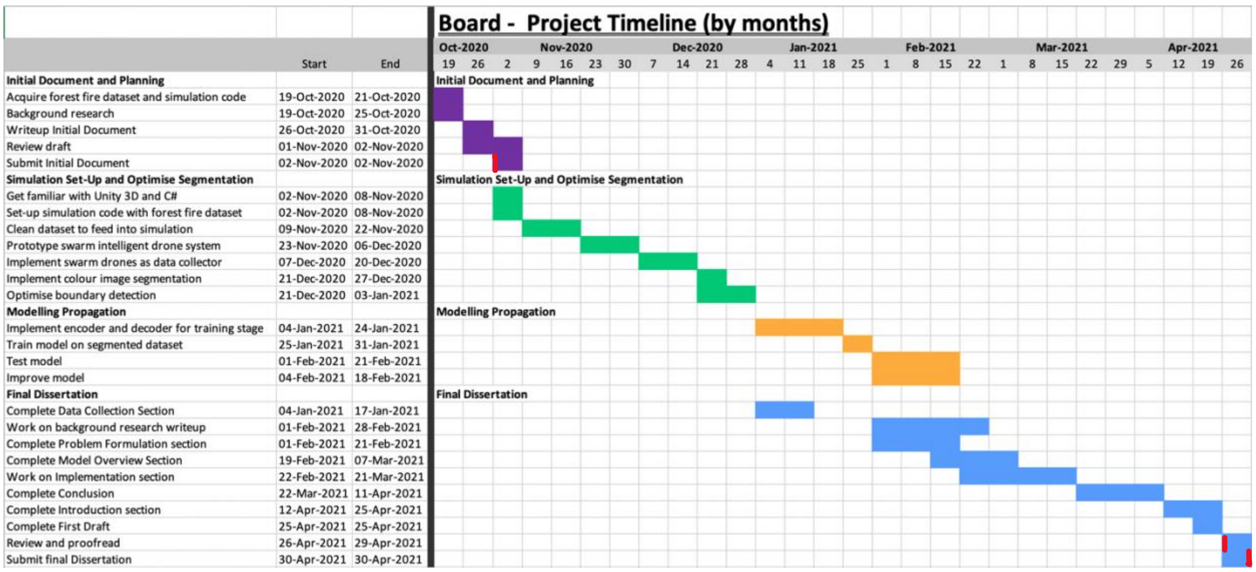

Figure 9: Project timeline

5 Conclusion

The objective of this project, as I have defined above, is to model the propagation of forest fires though simulation in Unity 3D, using SI for optimisation and ML techniques for training and modelling. Throughout this project I will be making use of new languages, libraries and frameworks. My goal is to achieve my primary objective whilst producing robust, high-quality code, that adheres to professional software engineering standards. As outlined in my project timeline above, I aim for the project to be completed in Spring 2021.

6 References

[1] M. Slezak, “3 billion animals killed or displaced in Black Summer bushfires, study estimates,” 20 July 2020. [Online]. Available: https://www.abc.net.au/news/2020-07-28/3-billion-animals-killed-displaced-in-fires-wwf-study/12497976. [Accessed 23 October 2020].

[2] D. F. K. Brulliard, “A Billions Animals Have Been Caught in Australia's fire's. Some May Go Extinct.,” January 2020. [Online]. Available: https://www.washingtonpost.com/science/2020/01/09/australia-fire-animals-killed/. [Accessed October 2020].

[3] C. A. J. M. Rábade, “Social Impact of Large-Scale Forest Fires,” in *Proceedings of the Second International Symposium on Fire Economics, Planning, and Policy: A Global View*, 2008.

[4] Union of Concerned Scientists, “The Connection Between Climate Change and Wildfires,” 11 March 2020. [Online]. Available: https://www.ucsusa.org/resources/climate-change-and-wildfires. [Accessed 23 October 2020].

[5] L. Q. Y. S. Y. S. S. C. J. Z. J. Yang, “Swarm Intelligence in Data Science: Applications, Opportunities and Challenges,” in *International Conference on Swarm Intelligence*, 2020.

[6] D. Schroeck, “Using Simulations To Train and Test Machine Learning Applications,” Transentis Consulting, March 2019. [Online]. Available: https://www.transentis.com/using-simulations-train-test-machine-learning-applications/. [Accessed October 2020].

[7] M. B. M. Dorigo, “Swarm Intelligence,” 28 September 2007. [Online]. Available: http://www.scholarpedia.org/article/Swarm_intelligence#Artificial.2FEngineering:_Swarm-based_Data_Analysis. [Accessed September 2020].

[8] K. M. P. Y. Liu, *Swarm Intelligence: Literature Overview,* Columbus, Ohio, 2000.

[9] F. C. B. Khaldi, “An Overview of Swarm Robotics: Swarm Intelligence Applied too Multi-robotics,” *International Journal of Computer Applications,* vol. 126, no. 2, 2015.

[10] J. V. K. P. A. Banks, “Particle Swarm Guidance System for Autonomous Unmanned Aerial Vehicles in an Air Defence Role,” *The Journal of Navigation,* vol. 61, no. 1, pp. 9-29, 2008.

[11] V. C. L. B. S. Gomes de Almeida, “Particle Swarm Optimization: A Powerful Technique for Solving Engineering Problems,” 3 December 2019. [Online]. Available: https://www.intechopen.com/books/swarm-intelligence-recent-advances-new-perspectives-and-applications/particle-swarm-optimization-a-powerful-technique-for-solving-engineering-problems. [Accessed October 2020].

[12] H. S.-M. G. Z. Z. Mohammad-Hossein Golbon-Haghighi, “Pattern Synthesis for the Cylindrical Polarimetric Phased Array Radar,” *Progress In Electromagnetics Research M,* vol. 66, no. N/A, pp. 87-98, 2018.

[13] M. Imran, “An Overview of Particle Swarm Optimization Variants,” in *Malaysian Technical Universities Conference on Engineering & Technology 2012, MUCET 2012*, Kuala lumpur, 2012.

[14] C. Ferguson, “Youtube: Particle Swarm optimisation,” Churchill Compsci Talks, 24 March 2018. [Online]. Available: https://youtu.be/DzcZ6bP4FGw. [Accessed 01 November 2020].

[15] Wikipedia , “wikipedia: Boids,” n.d n.d n.d. [Online]. Available: https://en.wikipedia.org/wiki/Boids. [Accessed 02 November 2020].

[16] A. v. Belle, “Color Image Segmentation by Particle Swarm Optimsation,” *Radboud University Nijmegen - Bachelor Thesis,* vol. n.a, no. n.a, pp. 0-75, 2012.

[17] Google, “Google Developers: Clustering In Machine Learning,” Google , [Online]. Available: https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages. [Accessed 01 November 2020].

[18] A. Ligmann-Zielinska, Spatial Optimization, Michigan: John Wiley & Sons., 2017.

[19] P. Z. Haghian, “Deep Representation Learning and,” *n/a,* vol. n/a, no. n/a, p. n/a, 2019.

[20] Microsoft, “Visual Studio | Best-in-class tools for any developer,” N.D.. [Online]. Available: https://visualstudio.microsoft.com. [Accessed October 2020].

[21] Microsoft, “Build Unity Games with Visual Studio,” N.D. [Online]. Available: https://visualstudio.microsoft.com/vs/unity-tools/. [Accessed October 2020].

[22] Unity, “Get to grips on the latest AI power in Unity,” N.D.. [Online]. Available: https://unity3d.com/how-to/unity-machine-learning-agents. [Accessed October 2020].

[23] C. d. l. T. B. Achtman, “ML.NET: Machine Learning for .NET Developers,” Code Magazine, N.D.. [Online]. Available: https://www.codemag.com/Article/1911042/ML.NET-Machine-Learning-for-.NET-Developers. [Accessed October 2020].

[24] S. McCarthy, “2020 Forest Fires Globally Could be Worse Than WWF Warns,” August 2020. [Online]. Available: https://www.worldwildlife.org/press-releases/2020-forest-fires-globally-could-be-worse-than-2019-wwf-warns. [Accessed October 2020].

[25] P. L. J. W. J. Y. Q. Cui, “Brief analysis of drone swarms communication,” in *2017 IEEE International Conference on Unmanned Systems (ICUS)*, Beijing, 2019.