Optimising Tree Planting Strategies and Visualising Layouts With Virtual and Mixed Reality

Sean Bochman

Master of Science

Computer Science

School of Informatics

University of Edinburgh

2024

Abstract

This skeleton demonstrates how to use the `infthesis` style for MSc dissertations in the School of Informatics. It also emphasises the page limit and associated style restrictions for Informatics dissertations with course code `INFR11077`. If your degree has a different project course code, then it is likely to have different formatting rules. The file `skeleton.tex` generates this document and should be used as a starting point for your thesis. Replace this abstract text with a concise summary of your report.

Research Ethics Approval

Instructions: *Agree with your supervisor which statement you need to include. Then delete the statement that you are not using, and the instructions in italics.*

Either complete and include this statement:

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: ???
Date when approval was obtained: YYYY-MM-DD
[If the project required human participants, edit as appropriate, otherwise delete:]
The participants' information sheet and a consent form are included in the appendix.

Or include this statement:

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Sean Bochman*)

Acknowledgements

Any acknowledgements go here.

Table of Contents

Chapter 1

Introduction

1.1 Background

At first thought, the importance of trees can be overlooked. However, trees provide immense benefits both environmentally and for human well-being [16, 27]. On people, trees in urban environments have been shown to improve focus and attention, increase people's ability to handle stress and anxiety, and overall improve mental health [27, 16]. Environmentally, the addition of trees in urban areas helps regulate and cool temperatures, improves air quality, and muffles noise [16]. Furthermore, as shown in many articles [12, 21, 6, 7, 10], climate change persists as a significant danger to humans.

One of the major catalysts for climate change is the vast quantity of CO2 filtered in the air [10]. Fortunately, trees remove CO2 through the process of photosynthesis [10].

1.2 Overview

This project applies trees' multitude of social and environmental benefits to craft an optimisation algorithm to maximise CO2 absorption in tree planting strategies in urban development. Some previous work exists [17, 14, 26, 4] for optimising planting strategy, but each is applied in a unique case study. This makes it unreliable to compare the methods used. Therefore, an accurate reconstruction of the case study used by Choi and Lee [4] is used to compare their linear programming optimisation to the genetic algorithm, another popular optimisation tool. Additionally, it is difficult to visualise the planting strategies, as the number of trees used and large planting areas are relatively complex. Therefore, virtual and mixed reality is a useful medium to visualise complex

data [9] and allows users to comprehend the planting strategies most accurately.

This paper contains five total chapters. Chapter 2 will outline the current literature for planting optimisation, optimisation algorithms of linear programming and evolution, and data visualisation in virtual and mixed realities. Chapter 3 describes the process of recreating Choi and Lee's case study, the creation of the genetic algorithm, and the results of two planting scenarios, maximising CO2 absorption and minimising cost. To visualise planting strategies, the Varjo XR-3 headset is used to create a virtual and mixed reality application in Unity, effectively visualising an optimised CO2 absorption planting strategy of Sundial Garden in Inverleith Park, Edinburgh. Lastly, Chapter 5 concludes all findings and limitations and directs future work for both the planting optimisation and visualising planting strategies in virtual and mixed reality.

1.3 Significance

The significance of this project is twofold. One, comparing the genetic algorithm to Choi and Lee's linear programming solution allows us to compare methods of optimising planting strategies faithfully, which is missing in the literature. Furthermore, given the immense benefits of trees in urban development, identifying better methods is crucial both socially and environmentally.

The second is in the visualisation of complex data. It is difficult to comprehend the planting strategies in a 2-dimensional plane fully. This makes optimisation worthless if urban planners cannot visually approve planting strategies. Planting in large complexes is expensive; Choi and Lee's layout, which minimises cost, still amounted to $335,801 for one apartment complex [4]. It is crucial to visualise these layouts most effectively. The optimised layouts can be visualised most effectively using virtual and mixed reality. This is particularly valuable for planners and other stakeholders, as it allows them to visualize the planting layout's impact on a space through a mixed-reality overview and a full-scale virtual simulation.

Chapter 2

Literature Review

Three main areas in the literature will be explored to combine tree planting optimisation with the virtual and mixed reality of the resulting planting strategy. Firstly, optimising the tree-planting strategy is addressed to find the current best practices. Secondly, the optimisation method is investigated between linear programming and the genetic algorithm. Lastly, the effectiveness of visualising complex data in virtual and mixed reality is surveyed. Together, these elements are encapsulated by this project, and the best findings in the literature motivate decisions throughout.

2.1 Optimising Tree Planting

Given the benefit of trees to both the ecosystem and human health, planting strategy is the next logical step to optimising the benefits. A 2020 article by Lin recognised these benefits and assembled a framework to prioritise certain trees depending on the environment, tree, and human demographic information [14]. This built greatly upon existing tree priority schemes, which tend to be environment-specific, whereas Lin's framework accounts for environmental variability [14]. Lin utilised the i-Tree Landscape tool to build a tree priority protection index (PPI) [14]. i-Tree Landscape is a web tool that provides U.S. data on forest cover, risks and benefits, and other demographic data on human health. This tool was chosen as it allowed a flexible way to build locally relevant PPIs [14]. The PPIs are constructed from three categories: environment, tree, and human. Then, these are compiled using the equation specified in equation (2.1) [14].

$$PPI = (C_1 \times W_1) + (C_2 \times W_2) + (C_3 \times W_3) \tag{2.1}$$

In equation (2.1), $C_{1,2,3}$ are associated with the three categories, and $W_{1,2,3}$ are the associated weights of each category. Varying the weights is helpful for varied objectives [14]. Lin provided a framework that accounts for several important factors, and while it prioritised tree type and planting location for environmental factors, it does not specifically optimise CO2 absorption. Furthermore, Lin's framework allows you to vary the objective with different weights, but each objective leads to different results [14]. This leaves the door open for a framework to account for multiple objectives.

In 2021, Nyelele and Kroll's article responded with a multi-objective framework for prioritising tree planting locations in urban areas. Nyelele and Kroll introduced the need for multi-objectives to combat the limitations of multiple constraints such as budget [17]. The framework is then optimised using linear programming [17]. 14 different scenarios were run as a case study in the Bronx, New York. The multi-objective function was tweaked. However, the budget of 400 million dollars remained constant [17]. Each scenario maximises a different singular objective while adhering to the other objectives' constraints [17]. This results in an optimised planting strategy depending on the weights and objective to optimise.

Another 2021 article produced a framework consisting of three parts: identification of native trees, selection of large-scale native tree planting locations, and lastly heightening awareness of the link between climate change resilience and human health [10]. Using Houston, Texas as a case study, native tree species were identified and subsequently ranked based on a combination of CO2 absorption, other air pollutants absorption, flood mitigation, and Urban Heat Island Reduction effects [10]. To determine location, sites that were experiencing disproportionately adverse health effects of climate change were selected. These effects consisted of areas with high rates of cardiac arrest and asthma attacks [10]. Lastly, the health department then worked with the local environmental group to raise awareness about the project and its positive effects [10].

The results of Hopkins's research ranked 54 native trees to an optimised subset of 17 trees. Houston saw large success in heightened awareness and a planting goal of 4.6 million trees by 2030 [10]. While Hopkins provided an optimal subset and identified urgent locations, it does not inherently optimise the specific planting locations in the urgent locations. This means that while the benefits to climate, environment, and humans are positive, they are not guaranteed to be optimised in the same way Nyelele and Kroll's linear programming solution does.

An article by Wang et al. in 2022 sought to determine how tree species should be selected for urban areas as well as which planting pattern of dense or sparse performs

best [26]. The River Delta in Hong Kong was chosen as a case study to investigate these questions. Wang et al. describe their model in detail, including elements of the city's energy, vegetation energy, moisture budgets, and the effects of the drag force of trees on the wind environment. To the effect of cooling, the sparse planting yielded a greater reduction than dense planting [26]. This provides important context to planting strategy. However, this article's relevance ends there. Wang et al. produced a model to conclude that tree species with larger crown diameters have greater cooling effects, especially when coupled with a sparse planting strategy [26]. This does not necessarily investigate the number of different configurations of tree planting but provides a general guide on overall strategy. This does not suggest an optimised strategy since that would be specific to a location.

Most recently, as of October 2022, Choi and Lee published an article on tree planting optimisation in urban residential spaces, noting the benefits to both the environment and human happiness [4]. Using an apartment complex in Seoul, South Korea, as a case study, Choi and Lee employ a linear programming solution [4] like Nyelele and Kroll. In this example, many constraints were considered, such as the minimum proportion of native trees, large trees, and evergreens. Other constraints were also incorporated, like cost and minimum and maximum canopy coverage [4]. Two scenarios were optimised through these constraints: total tree CO2 absorption and species diversity, and one minimisation scenario for total cost [4].

In this case study, Choi and Lee showcase the detail their model could incorporate through the constraints, such as planning certain areas to have large trees around the entrance and a hedge zone to act as a natural border to the property from the roads [4]. This article provides the most detailed specifications of the constraints used and information about the apartment complex compared to previous articles. Despite this, it is impossible to compare the results of Choi and Lee to the likes of Nyelele and Kroll or Lin, as each uses different constraints and locations. This proves to be a significant gap in the literature comparing optimisation methods.

Furthermore, while Wang et al. and Hopkins are valuable additions, providing further context to this topic, they serve more as a general rule of thumb rather than attempting to find a global maximum in the same way Chio and Lee or Nyelele and Kroll do. Lastly, both Choi and Lee and Nyelele and Kroll use a linear programming solution, but the literature lacks exploration of other optimisation strategies. A way to accurately compare methods and different strategies must be investigated to determine the best planting strategy for a location.

2.2 Genetic Algorithm and Linear Programming

The choice of the optimization algorithm is crucial to deducing a preferred planting strategy. Both Choi and Lee and Nyelele and Kroll utilised a linear programming optimization strategy; however, this review will also explore the genetic algorithm. A 2020 article by Katoch details the past and present of the genetic algorithm.

The genetic algorithm is an evolutionary algorithm designed to mimic the Darwinian theory of survival of the fittest. In the classical genetic algorithm there is a population of possible optimisations where each optimization has a chromosome, specifically defined by the context of the optimization, and represents the current state of the solution space [11]. The chromosomes are manipulated through genetic operators and scored using a fitness function. The best result of the population becomes the new population for the next generation [11]. The select operator repeatedly chooses chromosomes to update the population in the context of the fitness function. Crossover operates by taking two chromosomes and randomly choosing genes from each parent to produce a new chromosome. Lastly, the mutate operator randomly chooses a subset of genes in a chromosome and changes it [11].

The genetic algorithm has variations, such as the binary-coded genetic algorithm. In this case, chromosomes are binary [11]. This variant, however, has two issues. Firstly, the method has a high computational complexity [11]. More importantly, in the context of planting strategy, this would work for choosing a binary of to plant or not to plant but is limited by deciding on what tree to plant as this requires many options.

Linear programming is a relevant choice for optimising planting layouts. While this is a reasonable choice for linear and simple solutions, complicated locations with many constraints and tree types may not be solved optimally with a linear approach. Unsurprisingly, the introduction of non-linear terms increased computational complexity [3].

One method is to use transformations to linearise non-linear term(s). For example, to linearise a term of binary variables $x_i \times y_j$, replace it with an additional binary variable $z_{ij} = x_i \times y_j, \forall i \in 1,...,m, \forall j \in 1,...,n$ [3]. Asghari also describes other functions such as the square root function, maximum and minimum, and multiplication of continuous variables similarly to linearise [3]. Through these transformations, non-linear terms can be linearised to improve computational complexity and remain relevant to a linear programming solution while adhering to the complex nature of finding a global maximum for planting strategy.

A recent 2023 article dives deeper into the advantages of the genetic algorithm when applied to optimisation problems. Firstly, genetic algorithms do not require much math specific to the context of the problem. Therefore, the magic is its evolution and doesn't need to be constrained so heavily [8]. Additionally, genetic algorithms are adaptable to any problem, whether linear or nonlinear and discrete or continuous [8]. This makes it a clear contender for planting strategy, as it can handle many constraints without becoming so overbearing that a global solution becomes impossible to find.

Secondly, the genetic algorithm is robust in using the select, crossover, and mutation operators. This makes it incredibly effective in finding a global maximum or minimum [8].

Lastly, flexibility has also been an asset of the genetic algorithm. This means the algorithm can be improved with domain-specific heuristics to improve its computational complexity and ability to find a global maximum and minimum [8]. The genetic algorithm has several advantages in handling complex situations and effectively optimising compared to other optimisation methods. For this reason, the genetic algorithm should be explored further in the context of planting strategy. Given large areas, complex boundaries, and many factors to consider when choosing a tree to plant, the genetic algorithm can achieve a better optimisation than the linear programming solution that Choi and Lee utilised with its adaptability and robustness.

2.3 Virtual Reality/Mixed Reality Data Visualisation

Virtual reality is a technology that has yet to reach its full potential. Upgraded headsets with higher resolutions and features like eye tracking are being released yearly, such as with the Meta Quest 3 and Apple Vision Pro in 2023 [22]. Virtual reality's immersiveness has also been utilised for visual data analytics and visualisation [9].

A 2020 article introduced DataHop, a virtual reality data visualisation tool equipped with a filter to modify data [9]. To navigate around, Hayatpur et al. describe the lack of agreement in the literature. Proposals such as jumping, teleportation, and flying have all been suggested; however, for DataHop, they settled with a World-In-Miniature technique, which allows users to manipulate a small model of the environment [9]. A controller was used to interact with the displays [9].

To test the usability and usefulness of the visualisation software, a small study of six participants, two of which are females and aged between 22 and 39, evaluated the software [9]. Each participant was given an introduction, a video tutorial, and two

guided scenarios to use the software's key features. Afterwards, they were interviewed with high-level questions, and a set of 7-point Likert Scale questions about the usability and usefulness of DataHop [9]. While the users found the software helpful, there were concerns about the vastness of the environment, which could become disorienting [9]. This paper shows the usefulness of virtual reality environments for visualising complex data sets of higher dimensions. However, it lacks a broader spectrum of opinions. Six participants are not enough to accurately assess the usefulness of DataHop. The only criteria for the study was to own an Oculus headset [9], and they self-rated their data visualisation expertise [9]. This means it is still unknown what different expertise groups generally believe about DataHop without a larger sample size.

Mixed reality is another form that can be used for data visualisation. Although less common in the literature on virtual reality, an early 2022 article looked at mixed reality for healthcare [20]. In this, 3D annotation and medical visualisation are explored by sifting through 170 papers, articles, and TED talks on mixed reality applications in healthcare, which was narrowed down to 30 [20]. In these 30 papers and articles, 26% found the integration enhanced clinical decision-making for patient diagnosis and treatment [20]. 25% of the material found mixed reality supported learning and skill development for healthcare trainees, and finally, 45% indicated that most healthcare professionals were open to adopting mixed reality tools [20]. It is important to note that the percentages merely show that the article or paper indicated, and if it didn't, it doesn't necessarily mean that the opposite viewpoint was held, but rather that the article did not explore or mention it. While this is specific to healthcare, it does show that mixed reality has the potential for data visualisation, and the benefits seen in healthcare can be abstracted to other contexts.

While Hayatpur et al. showcase the usefulness of virtual reality for data visualisation, it was still unclear to how it would compare to a physical version. In 2022, however, Danyluk et al. explore this comparison [5]. Participants used small physical 3d bar charts and 2D and 3D on-screen versions to complete data analysis tasks in their study. the virtual environment was set up in Unity and opted to use the Vive controllers instead of hand-tracking, such as the Leap Motion controller, because of its lower learning curve [5]. The tasks to be completed consisted of indicating a range of values for a country, sorting values for a year, and locating three specific country-year pairs. Nine participants used the virtual hand-scale, table-scale, and room-scale visualisations and completed the tasks [5]. In the virtual, there was no difference between the hand scale and table scale, but the room-scale led to considerably slower times [5].

The physical portion was set up similarly, using physical objects instead of the virtual. Eighteen total participants through a university were recruited [5]. They found that error rates were generally the same compared to the virtual environment. However, the time spent on tasks differed for the scale and order of tasks in favour of the physical hands [5]. This article provides an interesting context for comparing the physical and virtual environments. However, there are several issues with their study choices. Firstly, it is unclear why they recruited different people for the virtual and physical components. Especially given the small sample sizes in both components, it is unfair to assume that the timings of tasks are representative of the population and thus comparable. Secondly, while they chose the controllers due to their low learning curve, they introduced a confounding variable. Instead of strictly comparing the physical and virtual environments, it is unclear if the difference is between the controller versus the hand, the physical versus the virtual, or a combination of the two.

Most recently, a 2023 article surveys the current literature of virtual data visualisation. One method of interpreting complex data is in information visualisation of spatial mappings [13]. Information visualisation works by enhancing human cognition through mental models of information. Multiple sources can be combined into one to create an easily digestible visualization [13]. The different movement techniques previously mentioned by Danyluk et al. are mentioned, like jump, portal, and teleportation [13, 5]. To create the visualisations, Unity is overwhelmingly chosen, making up 37.9% of the total software [13]. Notably, issues with the current hardware are addressed, such as cybersickness, latency in tracking, and low refresh rates [13]. Cybersickness is when the user feels nauseated and can be partially caused by latency and refresh rates, which break the immersion [13]. There are ongoing issues with cybersickness, and Korkut et al. show the gap in the literature for a fix. Regardless, the spatial mapping component is particularly relevant to tree planting strategy. Given its use in virtual data visualisation, the immersive nature of virtual reality can prove helpful for users digesting spatial information more intuitively.

Chapter 3

Optimising Planting Strategy

The genetic optimisation algorithm built in this project will be compared directly to the linear programming solution Choi and Lee [4] gave. To do this, a faithful adaptation of the apartment complex in Seoul, South Korea, must be accurately recreated programmatically using the same constraints outlined by Choi and Lee. In this project, this was accomplished in two iterations. Due to initial time constraints, the first iteration is missing two constraints. These constraints were left out at the time, as they focused more on the aesthetic of the planting strategy. Fortunately, iteration two rectifies this by adding the last two constraints and rerunning the genetic algorithm. This allows us to compare the two iterations and the linear programming optimisation given by Choi and Lee.

Choi and Lee describe three scenarios: maximising CO2 absorption, minimising cost, and species diversity [4]. Iteration one details the algorithm and results of maximising CO2 absorption, while iteration two incorporates both maximising CO2 absorption and minimising cost. The species diversity scenario is not replicated in this project.

3.1 Iteration One

3.1.1 Genetic Algorithm Creation

Iteration one details the bulk of the algorithm creation and adaptation of the apartment complex programmatically using Python. Object-oriented programming was adopted to build the algorithm and adapt the apartment complex. The genetic algorithm also requires a valid starting solution. Therefore, a greedy algorithm was implemented to populate the initial starting population.

The planting area is an apartment complex in Seoul, South Korea. Figure 3.1 shows the apartment complex.

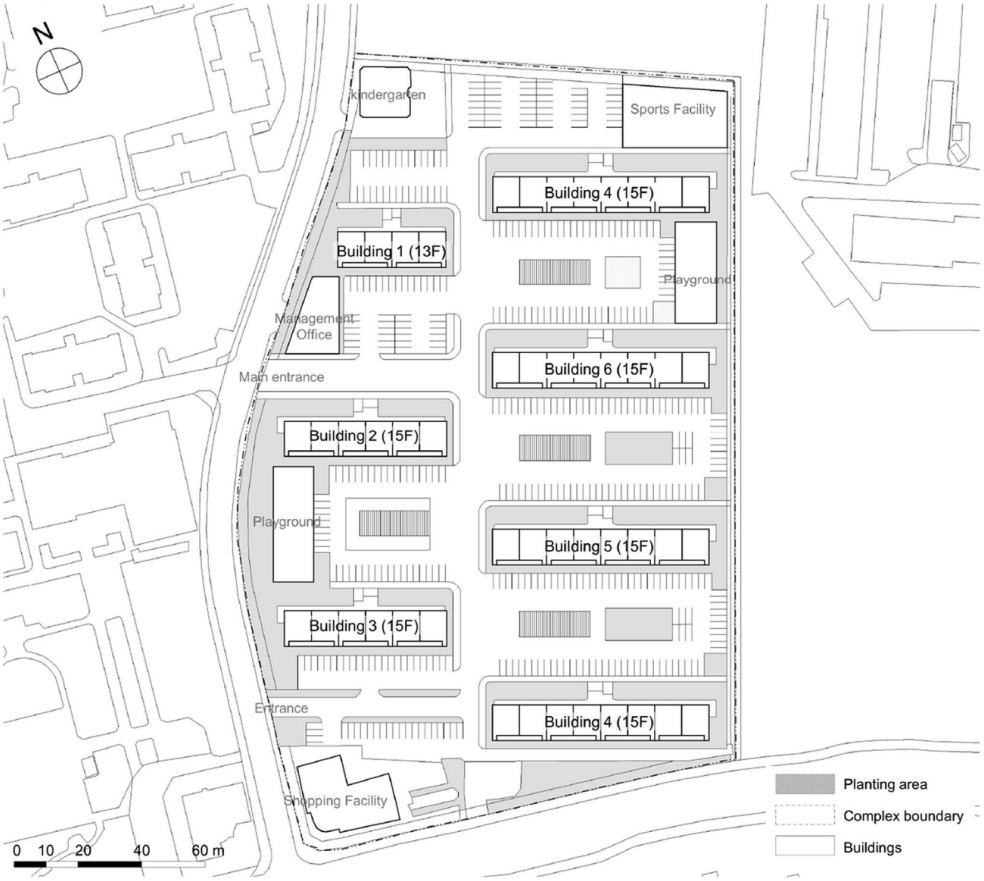

Figure 3.1: Apartment complex in Seoul, South Korea [4]

The grey area denotes the planting area. The border on the top and right edges are marked as hedge areas, where a screening tree must be placed, acting as a natural border [4]. Furthermore, the bottom and left sides are considered the roadside planting area. There is a pedestrian road moving upwards through the middle of the complex and between the buildings. Lastly, large tree planting areas are reserved at the entrances to the complex on the bottom and left side entrances [4]. It is this complex and the planting areas that must be adapted in Python for the genetic algorithm.

Choi and Lee also outline 21 different trees, shown in Figure **??**. The important features of a tree are its leaf type (Evergreen or Deciduous), credit count, CO2 absorption capacity (yearly), price, and whether it is native, large, or a screening tree.

No	Species	Leaf type	Plant size H(m)/W(m)/R(cm)	Credit	Crown area (m^2)	CO_2 absorbing capacity (kg/year)	Price ($ per tree)	Native tree[a]	Large tree	Screening tree
1	*Abies holophylla*	E	H3.0/W1.5/R7	1	1.8	2.2	135			●
2	*Pinus densiflora* (1)	E	H8.0/W5.6/R25	4	24.6	5.4	1,949		●	
3	*Pinus densiflora* (2)	E	H8.0/W5.6/R30	4	24.6	5.4	2,881		●	
4	*Pinus densiflora* var. globosa	E	H2.0/W2.5/R15	1	4.9	5.4	1,398			
5	*Taxus cuspidata*	E	H3.0/W2.0/R7	1	3.1	0.5	1,864			
6	*White pine*	E	H3.0/W1.5/R8	1	3.1	3.8	57			●
7	*Acer palmatum*	D	H4.0/W2.8/R20	4	6.2	3.1	796		●	
8	*Betula platyphylla*	D	H5.0/W3.5/R12	1	9.6	3.8	279			
9	*Cercidiphyllum japonicum* S. et Z	D	H4.5/W3.2/R15	2	8.0	3.8	423			
10	*Chaenomeles sinensis*	D	H4.0/W2.8/R15	2	6.2	3.8	381			
11	*Chionanthus retusus*	D	H4.0/W2.8/R15	2	6.2	3.5	550			
12	*Cornus officinalis*	D	H3.0/W1.5/R10	1	1.8	2.9	211			
13	*Ginkgo biloba*	D	H5.0/W3.5/R15	2	9.6	4.5	406	●		
14	*Kobus magnolia*	D	H3.5/W3.5/R15	2	9.6	3.8	423			
15	*Liriodendron tulipifera*	D	H5.0/W3.5/R15	2	9.6	3.8	389			
16	Oak	D	H4.0/W2.8/R15	2	6.2	3.8	423			
17	Persimmon	D	H3.5/W2.5/R12	1	4.9	3.8	186			
18	*Prunus armeniaca*	D	H2.5/W1.8/R6	1	2.5	3.8	55			
19	*Prunus yedoensis*	D	H4.5/W3.2/R15	2	8.0	9.0	576			
20	*Sophora japonica*	D	H4.5/W3.2/R15	2	8.0	3.8	322	●		
21	*Zelkova serrata*	D	H5.0/W3.5/R30	4	9.6	14.8	1,949		●	

e evergreen tree, *d* deciduous tree, *h* tree height, *w* crown diameter, *r* diameter of root

[a]Native tree can be used as a road side tree

Figure 3.2: Trees considered for planting of the apartment complex. This figure is given by Choi and Lee [4]

3.1.1.1 Constraints

The constraints are the last information necessary before building the genetic algorithm. The constraints largely stay the same between the two scenarios replicated in this project. However, this section notes a couple of differences. Importantly, the constraints covered here are implemented for iteration one, which is missing two of the constraints used by Choi and Lee. Section 3.2.1 details the final two constraints added in iteration two.

$$\sum \text{credits}_E + \sum \text{credits}_D \geq 0.2 \times A_L \tag{3.1}$$

$$\sum \text{credits}_E \geq 0.2 \times (\sum \text{credits}_E + \sum \text{credits}_D) \tag{3.2}$$

$$\sum \text{credits}_D \geq 0.2 \times (\sum \text{credits}_E + \sum \text{credits}_D) \tag{3.3}$$

$$\sum \text{credits}_{NE} + \sum \text{credits}_{ND} \geq 0.1 \times (\sum \text{credits}_E + \sum \text{credits}_D) \tag{3.4}$$

$$\sum \text{count}_{LE} + \sum \text{count}_{LD} \geq 0.06 \times (\sum \text{count}_E + \sum \text{count}_D) \tag{3.5}$$

$$\sum \text{canopy}_E + \sum \text{canopy}_D \leq 0.6 \times A_L \tag{3.6}$$

$$\sum \text{canopy}_E + \sum \text{canopy}_D \geq 0.4 \times A_L \tag{3.7}$$

$$\sum \text{count}_E \geq 0.015 \times (\sum \text{count}_E + \sum \text{count}_D) \tag{3.8}$$

$$\sum \text{count}_D \geq 0.015 \times (\sum \text{count}_E + \sum \text{count}_D) \tag{3.9}$$

$$\text{Length}_{SR} \leq \sum \text{crown\_width}_H \tag{3.10}$$

$$\text{planting\_area} \geq \frac{\text{tree\_width}}{4} \tag{3.11}$$

$$\sum \text{cost}_E + \sum \text{cost}_D \leq \$530000 \tag{3.12}$$

3.13: Constraints given by Choi and Lee [4] and implemented in iteration one Scenario One (maximising CO2 absorption).

Equations 3.1 through 3.11 are the constraints implemented in scenario one. The objective function:

$$\text{Maximise} \sum \text{CO2}_E + \sum \text{CO2}_D$$

maximises the CO2 absorption of the trees planted in the complex. ”credits” refers to the quantity credits, which is specific to a tree and given in Figure 3.2 of all tree information. The subscripts $_E$, $_D$, $_{NE}$, and $_{ND}$ refer to evergreen, deciduous, native evergreen, and native deciduous. $_{LE}$ and $_{LD}$ denotes the large evergreen and deciduous trees, while $_{SR}$ denotes the screening road. Lastly, $_H$ refers to the hedge planting zone, and A_L is the total apartment landscape area, set to 7326 m^2 [4].

Equation 3.1 constrains the minimum credit count to the total landscape area. Equations 3.2 and 3.3 ensure a minimum credit count of evergreen and deciduous trees.

Then, the minimum credit counts for native trees to total credit counts are given by equation 3.4. Equation 3.5 takes the count of large trees to be greater than the count of evergreen and deciduous trees and multiplied by the constant 0.06. Equations 3.6 and 3.7 denote the evergreen and deciduous trees' maximum and minimum canopy coverage. Furthermore, the minimum count of evergreen trees and the minimum count of deciduous trees are given in equations 3.8 and 3.9. The hedge area, expressed by the summation of each tree's crown width in the hedge zone, must be greater than the length of the screening road in meters. Lastly, the minimum planting area of a tree is given by equation 3.11 and the maximum cost by equation 3.12.

3.1.1.2 Genetic Algorithm - Object Oriented Programming

Several classes were created to program the apartment complex and genetic algorithm. Notably, the apartment complex required two important classes.

Early in the project, it was decided that the apartment complex would be depicted as a 2D grid. Each cell in the grid represents a half-meter by half-meter square area. The smaller the representation of each cell, the larger the grid becomes. This directly affects the time it takes to run the genetic algorithm. However, increasing the representation size, for example, to one meter by one-meter square area, limits the number of tree placement strategies. Furthermore, it directly affects the accuracy of the minimum tree planting area. Each tree has a specific radius to be free of another tree. These radii are rarely a whole number and thus can take up more cells than is necessary. For example, a radius of 1.5 meters would be depicted as 2 meters when each cell is considered one meter by one meter. However, the radius can be most accurately depicted using a half-meter by half-meter square area instead.

The *Grid* class handles the creation of the grid. In each class, there is both a class representation and a numerical representation. The class representation populates the grid with *Square* objects. The *Square* class specifies information about that particular planting location (cell), such as if it's plantable, a hedged area, the tree currently planted there if any, and more. On the other hand, the numerical representation serves as the chromosome for the genetic algorithm and shows as 0 for not planted or the tree number representing the tree type. Together, these hold the information on the grid and the relevant information necessary when testing different planting strategies.

The genetic algorithm also required three significant classes. To help build the algorithm, the deep Python library was utilised; however, many of the functions were overridden to be specified in the context of this project. Therefore, the *CustomGenet-*

icChanges class creates the initial population, defines the mate and mutates functions, and runs the algorithm. The deap's library select function was left unchanged. To mate two chromosomes, a random point was selected between 0 and the size of the flattened 2D array. Then, one offspring becomes the combination of the parents'. Offspring1 is parent1 up to the random point, and then parent2 from the random point to the end of the array. Offspring2 is simply the vice versa of offspring1.

The mutate function is more straightforward, having a random chance of 5% to either plant a random tree or swap out the current tree in a cell.

The *AlgorithmMutations* class handles planting and swapping a currently planted tree for a new one. When the mutate function tries to plant a tree at a specified cell, the cell and tree type are given to *AlgorithmMutations* to determine if the tree can be planted. This utilises the information in the Grid class to determine if the spot is plantable. Furthermore, it first checks each spot in the tree's radius to ensure each spot is plantable. This prevents planting a tree from being too close to another tree. This is enough to run the genetic algorithm. However, it can be improved by a slight change in the planting logic. Currently, if a tree can almost be planted, but a couple of cells are overlaid with another tree, it is considered unplantable. However, a quick local search of a 10 square meter box around the current planting spot can help find a valid planting spot a more significant percentage of the time. This also helps lead to more observed strategies and a greater chance of finding a global maximum or minimum.

Lastly, the constraints are represented in the *ScenarioOneConstraints* class. The class acts as the fitness function for the genetic algorithm, looping through the inputted grid and tracking the values for each constraint. To maximise CO2 absorption, a value of 0 is returned if a constraint is violated. Otherwise, the total CO2 absorption is returned.

Scenario one maximises CO2 absorption. Therefore, the weight of the fitness scores is set to 1.0. The weight is applied in the "Fitness Base" function defined by the deap library. This helps sort the fitness scores received from the fitness function defined by the *ScenarioOneConstraints* class for the select function to choose from for the next generation population. The mutate function has a random chance of 5% to mutate on any given cell. The tree selected is entirely random. A random seed of 100 is set for reproducible results. There is a 50% chance to apply the crossover function to a chromosome and a 20% chance to apply the mutate function to a chromosome. Finally, the resulting offspring of the population is sent to the *ScenarioOneConstraints* class to evaluate and determine its fitness and select the best offspring to become the new population.

3.1.1.3 Picture to NumPy Array

Before the grid can be created based on the apartment complex and run the genetic algorithm described in section 3.1.1.2, the apartment complex must first be accurately recreated programmatically. Given the large area of the apartment complex (33,200 m^2) coupled with the decision for each cell to represent a half-by-a-half meter squared area, this would result in more than 66,400 cells in the 2D array when accounting for the extra white space around the apartment complex. It is not feasible to hardcode the plantable areas, necessitating a faster approach. To accomplish this, the original picture shown in Figure 3.1 is coloured to differentiate the different planting areas, as shown in Figure 3.3. The colouring was done using the GoodNotes app on an Apple iPad. This image was also resized so that the $1m$ scale shown in Figure 3.1 is aligned with 10 pixels. In other words, every 10 consecutive pixels equates to $1m$. To accomplish this, Paint 3D by Microsoft was used to give pixel coordinates and the ability to change picture dimensions and resolutions [15]. The dimensions of the image that satisfied this scaling were 1675 x 2572. Note that the hedge planting areas are not coloured differently. Changing the plantable areas above a certain y value and a certain x value is simple to change to a hedge planting area. The reasoning for doing this is explained later in this section.

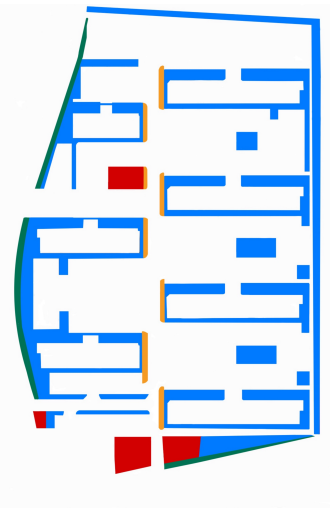

Figure 3.3: Apartment complex coloured. Blue represents plantable areas. Red is large tree planting, while green is roadside planting. Lastly, the orange depicts the pedestrian road planting.

With the coloured picture, a Jupyter notebook was created to greyscale the image and convert it to an array of pixel-greyscaled values. Furthermore, the current grid

has 5 times the number of cells as required. As every 10 pixels equals $1m$, this needs to be translated into the grid so that each cell represents a half-by-half meter square area. This is a straightforward task, where every 5x5 block was condensed into one block, where the value is the most common in that 5x5 block of cells. Table 3.1 shows the grayscale value for each of the colours. To validate that the resizing and pixel condensing are correct, random coordinates were selected around the original image provided in Figure 3.1 and converted by hand to coordinates that should appear in the finalised grid. Appendix Figure A.1 shows the circles aligning on the finalised grid from the initial image in Figure 3.1. From here, the grid of grayscaled values was saved and imported into the main Python project to initialise the Grid class.

Table 3.1: Colour To GrayScale Values

Colour	Blue	Red	Green	Orange
Grayscaled Value	101	63	77	168

Using the values, the Square class with the correct attributes could be initialised within each cell in the Grid object. However, the resulting grid contained some inconsistencies. The more colours in the grayscale, the more outputs there were for one colour. For example, blue could have cells filled with 101, 102, 98, and more. Cutting down on the number of colours could prevent some of these issues. The hedge area was chosen because of its position on the image. Therefore, all plantable areas above a x value of 316 and above a y value of 22 can be determined to be hedge planting areas. Additionally, further inconsistencies can be rectified by filling in gaps. For example, if a cell is labelled as unplantable, but the cells to the right, left, above, and below are plantable, then the cell's current cell must be a plantable area. This logic was applied to all planting areas. Figures 3.4 and 3.5 compare before and after filling in gaps.

Although not perfect, many of the inconsistencies are now rectified. Furthermore, the decision was made to undercompensate to prevent the possibility of overcompensating. This means that although the remaining gaps constitute less plantable area than Choi and Lee, it guarantees that the results are within the confines of the constraints and areas laid out by Choi and Lee.

Figure 3.6 shows the complete Grid class initialised. Red represents the hedge area, grey represents big tree planting, and blue represents the general plantable area.

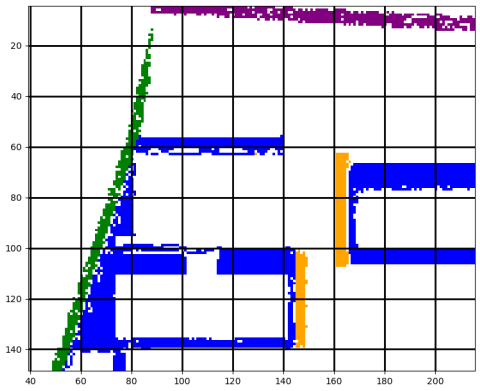

Figure 3.4: Grid class initialised before gaps filled in.

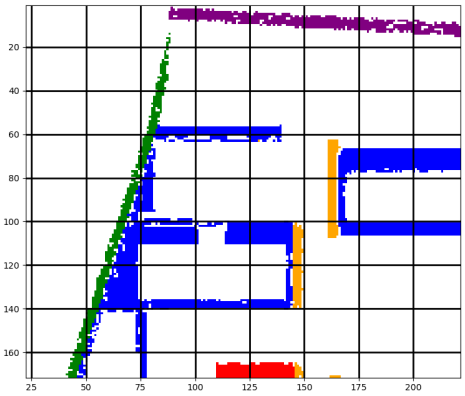

Figure 3.5: Grid class initialised after gaps filled in.

3.1.1.4 Greedy Algorithm

The last piece to run the genetic algorithm is a valid random starting state. With as many constraints as the expansiveness of the apartment complex, it is unreasonable to determine a starting place by human hand. Therefore, a greedy algorithm was constructed to return random starting states that confirm all constraints.

Given the several planting areas, these are first initialised in the grid. Each plantable hedge spot is planted with a random choice of the two screening trees, and the large tree areas are planted with large trees. Following this, the grid is looped through once more, now checking for each plantable cell. The *ScenarioOneConstraints* class determines which current constraint is being violated for scenario one. Given this, a tree is chosen randomly, which would help validate the constraint. For example, if the total credit count of evergreen were too few, then an evergreen tree would be chosen to plant in the current plantable cell. The first iteration that validates all constraints is immediately returned as a starting state for one chromosome in the population. Figure 3.7 shows an example of this greedy algorithm.

By planting in each plantable spot, the grid is hardly filled. This results in a severe aesthetic issue, negating the purpose of planting in an apartment complex and its effect on human happiness [27]. Therefore, tweaking the greedy algorithm can achieve a more aesthetic layout. To do this, random chance was incorporated into whether to plant. For hedge planting, there was a $\frac{2}{25}$ chance of planting a screening tree. A chance of $\frac{1}{25}$ chance was incorporated to plant a large tree. A $\frac{1}{40}$ chance was integrated on the last loop for the remaining plantable area. These changes resulted in Figure 3.8, a much more aesthetic starting chromosome and more relevant to optimising a planting strategy

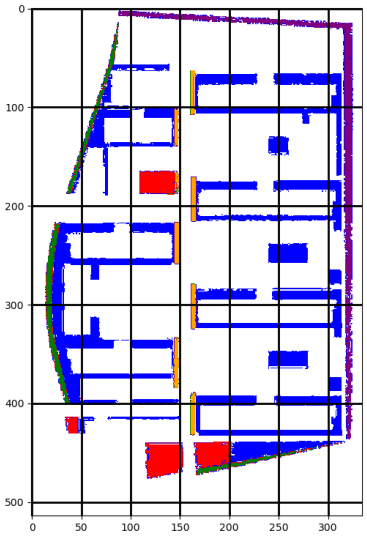

Figure 3.6: *Grid* class initialised with an adapted apartment complex from Seoul, South Korea.

in an apartment complex.

3.1.2 Scenario One

With all pieces finalised, the genetic algorithm is complete and runnable. The algorithm was run twice, once with a population size of 250 for 250 generations and the other with a population size of 500 for 50 generations. This allows for the comparison of population size to several generations, leading to more favourable results. Choi and Lee achieved a total CO2 absorption of 4,383.10 kg/year [4]. It is this value that the genetic algorithm aims to beat.

3.1.2.1 Results and Discussion

Appendix Figures A.2 and A.3 depict the population size of 250. In A.2, the average fitness of the generation is plotted over the generations, while each generation's best fitness is plotted in A.3. Both graphs show how quickly the fitness score converges, tapering off entirely after the 150th generation. The average fitness begins to oscillate after the 50th generation. Despite this, the best fitness still improves, and since a singular optimisation is sought, the oscillating nature of the population is irrelevant. Figure A.3 shows the fitness breaches over 4300 kg/year but falling just short of the

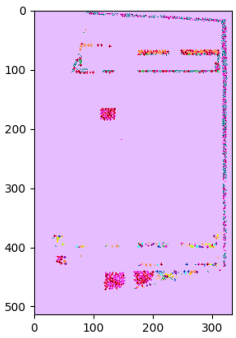

Figure 3.7: Valid starting state returned by greedy algorithm planting in each plantable spot.

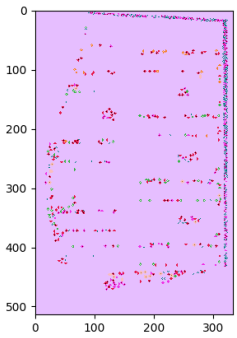

Figure 3.8: Valid starting state returned by greedy algorithm planting a specified percentage of the time in a plantable spot.

goal at a maximum of 4351.3 kg/year. The values are shown in Table 3.2.

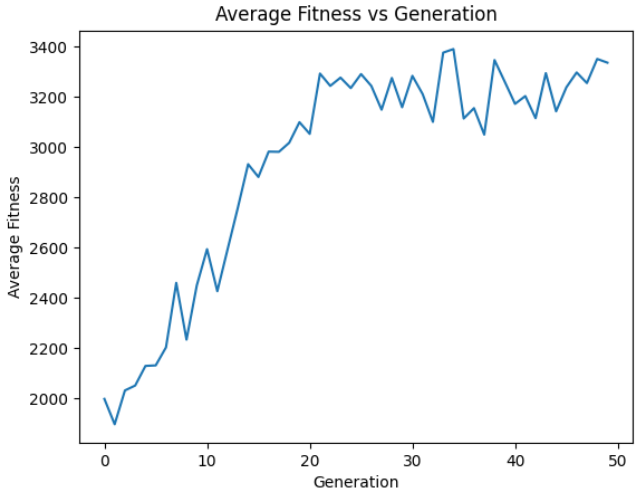

Figure 3.9: Average fitness score (total CO2 absorption) for a population size 500 for 50 generations.

Since the algorithm converges quickly, fewer generations are needed to optimise. Instead, a larger population size may help the algorithm explore more planting strategies and find a more optimal layout. Figures 3.9 and 3.10 show the average and best fitness scores for a population 500 run for 50 generations. The average fitness graph no longer oscillates, and instead, both the average and best fitness appear to be still improving by the 50th generation. This would suggest that a better optimisation is still achievable with more generations. Most importantly, an optimised value of 4535.3 is achieved, as shown in Table 3.2. This surpasses Choi and Lee's 4,383.10 kg/year by an additional

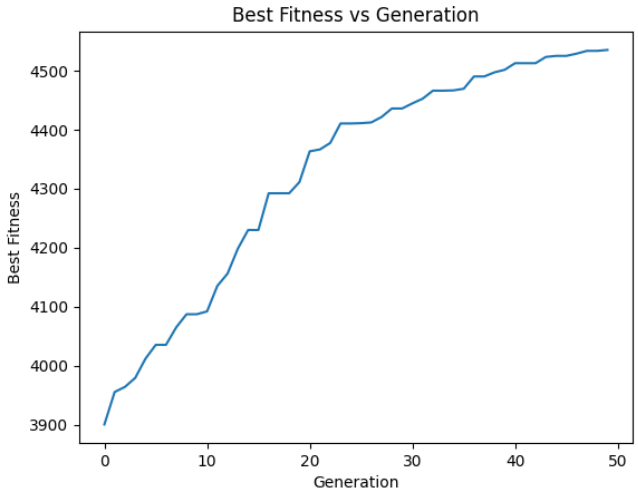

Figure 3.10: Best fitness score (total CO2 absorption) of a population size of 500 for 50 generations.

152.2 kg/year of CO2 absorption.

Table 3.2: Genetic Algorithm Iteration One Scenario One Results

Population Size	Generations	CO2 Absorption	Total Price
250	250	4351.3	$410,940
500	50	4535.3	$429,727

Table 3.2 details the total CO2 absorption for both population sizes and the total price to plant. In both population sizes, the total price is far below the maximum price set of $530,000. This is also significantly less expensive than Choi and Lee's planting strategy, which cost $477,792.

Furthermore, the three most planted trees in both population sizes are Abies holphylla, White pine, and Zelkova serrata. The first two are the screening trees with small crown widths, as shown in Figure 3.2. Therefore, these two should be planted heavily for the screening area. Zelkova serrata is also heavily planted due to its sizeable yearly CO2 absorption capacity of 14.8 kg/year. This tree is planted 104 times in the 250 population and 118 times in the 500 population. This means the Zelkova serrata alone is responsible for 1539.2 1746.2 kg/year of CO2 absorption, respectively. Choi and Lee have similar findings, with the Zelkova serrata being planted 146 times in their optimisation [4].

Overall, the genetic algorithm outperforms the linear programming solution Choi

and Lee gave in iteration one. The best planting strategy is depicted in Figure 3.11. The 250 population grid strategy is shown in the Appendix Figure A.4. These figures, unfortunately, do little to visualise the planting strategy. Some understanding of the overall fill throughout the apartment complex is given. However, it is impossible to tell what tree type is planted, where the spacing of trees is, or understand how the tree looks in size, bark, or leaf colour. Chapter 4 rectifies this issue by utilising the Varjo XR-3 to showcase the planting strategy in virtual and mixed reality, a more suitable medium for visualising complex data [9, 5].

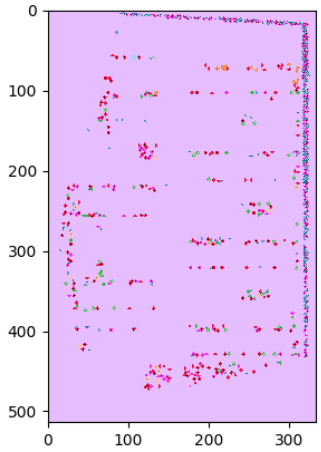

Figure 3.11: Planting strategy for the best grid of 500 population size.

3.2 Iteration Two

Iteration two rectifies the issue of the two missing constraints, and now, the roadside and pedestrian road planting constraints are being applied. Furthermore, this is used in both scenario one, maximising CO2 absorption, and scenario two, minimising cost. Section 3.2.1 discusses the final two constraints added to scenario one and details the constraints of scenario two.

3.2.1 Adding in Remaining Constraints

The last two constraints for scenario one are detailed in equations 3.14 and 3.15. Here, $_{RS}$ and $_{PR}$ stand for roadside and pedestrian road, respectively. $_{NRS}$ represents the native trees planted on the roadside. Equation 3.14 ensures a minimum count of native trees

on the roadside plantable area, and Equation 3.15 constrains the minimum count of trees on the pedestrian road.

$$\text{length}_{RS} \leq \sum \text{count}_{NRS} \tag{3.14}$$

$$\text{length}_{PR} \leq \sum \text{count}_{E} + \sum \text{count}_{D} \tag{3.15}$$

3.16: Constraints given by Choi and Lee [4] and implemented in iteration two Scenario Two (minimising cost).

For scenario two, minimising cost, the objective function is given by:

$$\text{Minimise} \sum_{\text{total\_cost}_E} + \sum_{\text{total\_cost}_D}$$

[4]. For this scenario, the constraints largely remain the same. However, equation 3.12 is removed and denotes the minimum CO2 absorption required instead. The constraints for scenario two are detailed in Equations 3.17 through 3.30. Note that equation 3.30 requires that a minimum CO2 absorption of all trees be greater than 3500 kg/year [4].

$$\sum \text{credits}_{E} + \sum \text{credits}_{D} \geq 0.2 \times A_L \tag{3.17}$$

$$\sum \text{credits}_{E} \geq 0.2 \times (\sum \text{credits}_{E} + \sum \text{credits}_{D}) \tag{3.18}$$

$$\sum \text{credits}_{D} \geq 0.2 \times (\sum \text{credits}_{E} + \sum \text{credits}_{D}) \tag{3.19}$$

$$\sum \text{credits}_{NE} + \sum \text{credits}_{ND} \geq 0.1 \times (\sum \text{credits}_{E} + \sum \text{credits}_{D}) \tag{3.20}$$

$$\sum \text{count}_{LE} + \sum \text{count}_{LD} \geq 0.06 \times (\sum \text{count}_{E} + \sum \text{count}_{D}) \tag{3.21}$$

$$\sum \text{canopy}_{E} + \sum \text{canopy}_{D} \leq 0.6 \times A_L \tag{3.22}$$

$$\sum \text{canopy}_{E} + \sum \text{canopy}_{D} \geq 0.4 \times A_L \tag{3.23}$$

$$\sum \text{count}_{E} \geq 0.015 \times (\sum \text{count}_{E} + \sum \text{count}_{D}) \tag{3.24}$$

$$\sum \text{count}_{D} \geq 0.015 \times (\sum \text{count}_{E} + \sum \text{count}_{D}) \tag{3.25}$$

$$\text{Length}_{SR} \leq \sum \text{crown\_width}_{H} \tag{3.26}$$

$$\text{planting\_area} \geq \frac{\text{tree\_width}}{4} \tag{3.27}$$

$$\text{length}_{RS} \leq \sum \text{count}_{NRS} \tag{3.28}$$

$$\text{length}_{PR} \leq \sum \text{count}_{E} + \sum \text{count}_{D} \tag{3.29}$$

$$\sum \text{co2\_absorption}_{E} + \sum \text{co2\_absorption}_{D} \geq 3500 \tag{3.30}$$

3.31: Constraints given by Choi and Lee [4] and implemented in iteration two Scenario Two (minimising cost).

3.2.2 Genetic and Greedy Algorithm Updates

Firstly, ScenarioOneConstraints is modified to incorporate the final two constraints, again returning 0 if the constraint is violated.

An additional class is made for scenario two to function. Scenario two minimises cost and works similarly to *ScenarioOneConstraints*. The difference is the *ScenarioTwoConstraints* class, which defines a slightly different fitness function. In this scenario, an arbitrarily large value of 99999 is returned if a constraint is violated. If all constraints are validated, then the cost of the planting strategy is returned. Additionally, a weight of -1.0 is used in the fitness score since we are minimising instead of maximising. Other than these two changes in the code, the rest of the program functionality is consistent between the scenarios.

Exactly as in iteration one, both scenarios have a mutate function set to a random chance of 5% to mutate on any given cell, a 50% chance to apply the crossover function, and a 20% chance to apply the mutate function to a chromosome.

Furthermore, with the introduction of the final constraints, the greedy algorithm also required a couple of changes. In the same way, the hedge and big tree areas are planted. First, the pedestrian road and road are planted before the rest of the general planting area. If the cell is a pedestrian road, there is a $\frac{3}{10}$ chance of planting. If it is a road, there is a $\frac{2}{25}$ chance of planting a native tree. The rest of the greedy algorithm and probability of planting remains consistent with iteration one. This algorithm now produces valid chromosomes to initialise the population for scenarios one and two.

3.2.3 Scenario One Results and Discussion

Scenario one, with all constraints defined by Choi and Lee, is run by the genetic algorithm in two ways. The first is with a population of 250 for 100 generations, and the second is with a population of 500 for 50 generations.

Figure 3.12 and Appendix Figure A.5 detail the average fitness score of each generation for the 250 and 500 populations. For the 250 population, the fast convergence seen in iteration one is also present here. However, the average fitness score for the 500 population is much more gradual. Figure 3.13 and the Appendix Figure A.6 show the best fitness score for each generation, making it easier to see for the 250 population how quickly it converges to its maximum. Compared to iteration one, the 250 population group slightly outperforms the 500 population. Despite this, both populations significantly outperformed Choi and Lee 4,383.10 kg/year strategy.

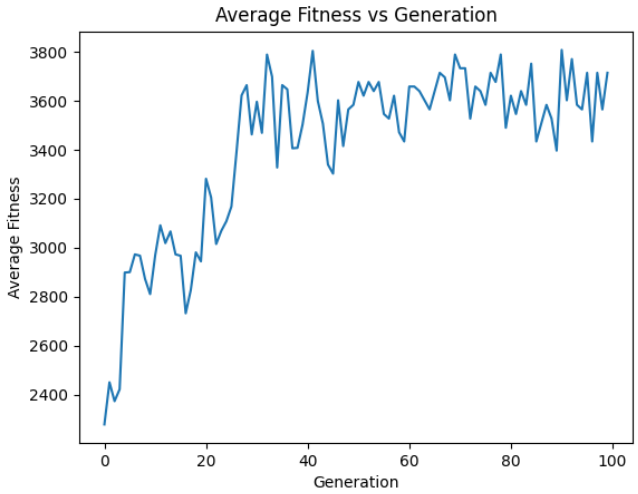

Figure 3.12: Average fitness score (total CO2 absorption) for a population size 250 for 100 generations.

Table 3.3 shows that while both populations improve from iteration one, the cost of the planting strategies has increased dramatically. Both are less than $2,000 away from the maximum cost, whereas the results from iteration one were far cheaper. Interestingly, while the 500 population is more expensive, it is 35.4 kg/year CO2 absorption less than the cheaper 250 population strategy. The same as in iteration one, the Zelkova serrata tree is a popular tree choice with 130 planted for the 250 population and 114 planted for the 500 population.

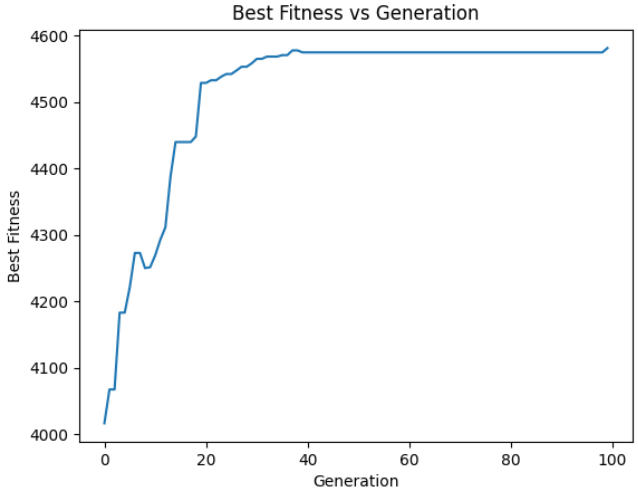

Figure 3.13: Best fitness score (total CO2 absorption) for a population size 250 for 100 generations.

Table 3.3: Genetic Algorithm Iteration Two Scenario One Results

Population Size	Generations	CO2 Absorption	Total Price
250	100	4581.5	$528,068
500	50	4546.1	$528,860

The planting strategy for the two population groups is shown in Figure 3.14 and Appendix Figure A.7. Notably, compared to Appendix Figure A.4 and Figure 3.11, the pedestrian road now has trees planted alongside it. This may explain the improvement in total CO2 absorption, as the Zelkova serrata tree, with its large CO2 absorption capacity, has more space to plant. Since it is a large tree with a crown width of 3.5m, detailed in 3.2, it requires more space to plant than the other trees. With the additional planting space forced to be planted by Equation 3.15, the Zelkova serrata can be planted more often. This is also shown as iteration one planted the Zelkova serrata 104 and 118 times for the 250 and 500 populations, respectively. However, it was planted 130 and 114 times for the iteration two 250 and 500 populations, respectively. The iteration two 250 population performed the best overall and planted the most Zelkova serrata trees. This tree alone isn't everything, as Choi and Lee planted the Zelkova serrata 146 times, yet are 198.4 kg/year of CO2 absorption, less than the 130 times it was planted in the 250 population strategy.

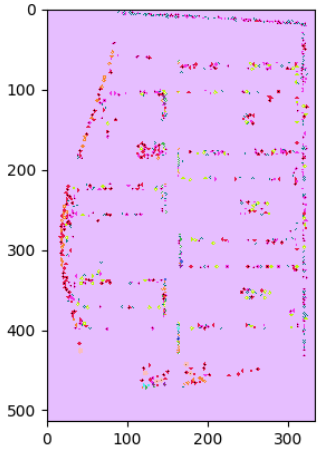

Figure 3.14: Iteration two scenario one best planting strategy from population size 250 for 100 generations.

3.2.4 Scenario Two Results and Discussion

Scenario two minimises the cost of the planting strategy while adhering to the constraints laid out. Notably, at least 3,500 kg/year of CO2 absorption must be obtained.

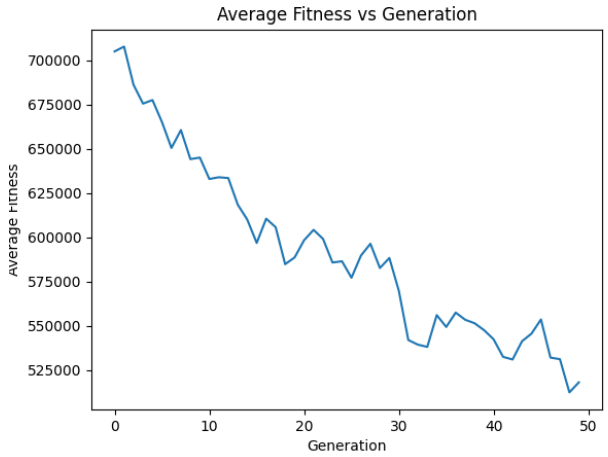

Figure 3.15: Average fitness scores for scenario two for 1000 population and 50 generations.

The genetic algorithm for scenario two minimises quickly for the 250 population as shown in the Appendix Figures A.8 and A.9. For the 500 and 1000 population, however, it converges much more slowly. The 500 population is shown in the Appendix Figures A.10 and A.11. The 1000 population is shown in Figures 3.15 and 3.16. The general trend is a better minimisation for higher population sizes, although the 1000 population may have improved past the 50 generations it was allotted.

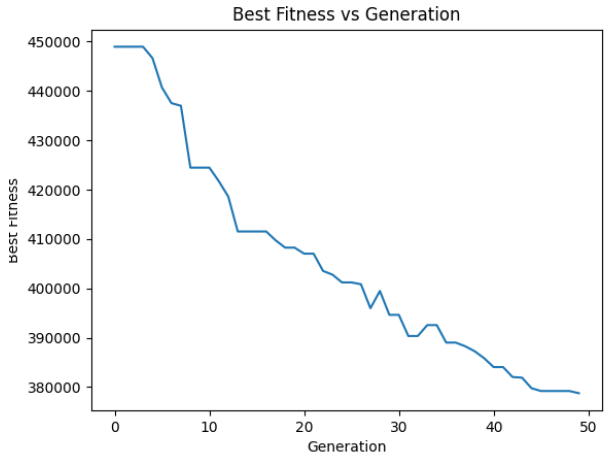

Figure 3.16: Best fitness scores for scenario two for 1000 population and 50 generations.

As shown in Table 3.4, the best minimisation came with the 1000 population at $378,738. Choi and Lee achieved a minimisation of $335,801, meaning the genetic algorithm is $42,937 above the linear programming solution. While the 1000 population may have still improved past 50 generations, it is unlikely to have closed that significant of a gap.

A grid search of different parameters on population sizes, number of generations, and chances of applying the mutate and mate functions may improve further upon the minimum found in Table 3.4.

Many constraints deal with minimum values, such as the minimum number of credits of a native tree. Since scenario one maximises and the greedy algorithm meets these minimums at initialisation, the exploration does not violate these constraints as often. However, by minimising cost, many chromosomes in the population will violate these constraints. Therefore, if a higher percentage of chromosomes in scenario two violates the constraints compared to scenario one, then less exploration is done. This is seen in the higher population achieving better minimisations when the same wasn't found in scenario one, as shown in Table 3.2. The higher population results in more valid chromosomes in each generation, allowing for greater exploration. This may explain the reasoning behind the genetic algorithms' success in scenario one and failure in scenario two in comparison to Choi and Lee.

Table 3.4: Genetic Algorithm Iteration Two Scenario Two Results

Population Size	Generations	CO2 Absorption	Total Price
250	100	3524.5	$419155
500	50	3504.1	$383,518
1000	50	3505.6	$378,738

The planting strategy for each population size is shown in the Appendix Figures A.12, A.13, and A.14.

Chapter 4

Visualisation with Varjo XR-3

While the genetic algorithm has proved to be a powerful optimisation tool for strategically planting complex layouts, it is difficult to visualise. The 2-dimensional nature of the graph, such as seen in Figure A.14, inhibits the ability to appreciate the distance between trees and size differences. Furthermore, a serious urban planner must understand how the trees would visually look in the space. To accomplish this, virtual and mixed reality headsets are capable of helping users visualise complex data more efficiently [9]. Virtual reality allows users to immerse themselves in the space and better understand how the optimised planting strategy would visually look. For the apartment complex, the cost of tree planting ranged from $378,738 to $528,860. It would be unreasonable to spend this without knowing and agreeing on the visual appearance of the planting strategy.

The Varjo XR-3 is a high-end virtual and mixed reality headset, which, at the time of release in December 2020, boasted the highest resolution and widest field of view of any other headset [2]. Furthermore, the XR-3 has integrated hand and eye tracking [2].

The purpose of this section is to most accurately visualise the optimised planting strategy obtained by the genetic algorithm within both a virtual and mixed-reality application using the Varjo XR-3 headset. In particular, the application should allow users to manipulate the tree layout, a necessary feature for urban planners to make planting strategies more aesthetically pleasing. Furthermore, the application was designed to fully leverage the capabilities of the XR-3 by eliminating dependency on a traditional controller. The entire interface is interactable through hand and eye gestures, providing a more immersive and intuitive user experience.

4.1 Edinburgh Inverleith Park

Sundial Garden, a segment of Inverleith Park, Edinburgh, was utilised to showcase this application. This provided a large enough area to demonstrate the effectiveness of the genetic algorithm while remaining small enough to demo the virtual and mixed-reality application. Figures 4.1 and 4.2 show the Sundial Garden.

Figure 4.1

Figure 4.2

A similar process described in Chapter 3 is used to initialise the Grid class for the Sundial Garden. Before running the genetic algorithm, a few changes were also made to the constraints, trees, and greedy algorithm. These differences are described in section 4.1.1.

4.1.1 Repurposing Genetic and Greedy Algorithms

The first step was to select trees capable of growing in Scotland. Although the trees selected by Choi and Lee are generally native to Asia, many of them can still grow to maturity in Scotland. Figure 4.1 depicts the remaining trees that can grow in Scotland. The ID number correlates with the number from 3.2. The Royal Horticultural Society (RHS) registry was consulted to validate that a tree could grow to maturity in Scotland [19]. Choi and Lee's trees are used because of the detailed tree information in their table, shown in Figure 3.2. The genetic algorithm is crafted along the tree features existing in this table and, therefore, made sense to use the trees in which this information is already known.

With a new space, some of the constraints outlined by Choi and Lee are no longer relevant. For example, Sundial Garden has no road or hedge planting areas. Therefore,

Table 4.1: Trees from Choi and Lee usable in Edinburgh, Scotland

ID	**Tree Name**
1	Abies Holophylla
5	Taxus Cuspidata
6	White Pine
7	Acer Palmatum
8	Betula Platyphylla
9	Cercidiphyllum Japonicum
11	Chionanthus Retusus
12	Cornus Officinalis
13	Ginkgo Biloba
14	Kobus Magnolia
15	Liriodendron Tulipifera
16	Oak
17	Persimmon
18	Prunus Armeniaca
20	Sophora Japonica
21	Zelkova Serrata

these constraints were removed. Equations 4.1 through 4.9 depict the finalised constraints for the genetic algorithm fitness function. P_L stands for the total park landscape area, valued at 23358 m^2. An additional constraint of a maximum of 45 trees is set in Equation 4.1. This differs from Choi and Lee, but it helps prevent the Sundial Garden from aesthetically looking like a forest and keeping space for people to walk, play, and enjoy the sunshine. The total canopy coverage area is also set to be low in Equations 4.5 and 4.6 to help with this similarly. Lastly, the cost of $50,000 was chosen.

$$\sum \text{count}_E + \sum \text{count}_D \leq 45 \tag{4.1}$$

$$\sum \text{credit}_E + \sum \text{credit}_D \geq 0.001 \times P_L \tag{4.2}$$

$$\sum \text{credit}_E \geq 0.2 \times (\sum \text{credit}_E + \sum \text{credit}_D) \tag{4.3}$$

$$\sum \text{count}_{EL} + \sum \text{count}_{DL} \geq 0.06 \times (\sum \text{count}_E + \sum \text{count}_D) \tag{4.4}$$

$$\sum \text{canopy}_E + \sum \text{canopy}_D \geq 0.008 \times P_L \tag{4.5}$$

$$\sum \text{canopy}_E + \sum \text{canopy}_D \leq 0.012 \times P_L \tag{4.6}$$

$$\sum \text{count}_E \geq 0.015 \times (\sum \text{count}_E + \sum \text{count}_D) \tag{4.7}$$

$$\sum \text{count}_D \geq 0.015 \times (\sum \text{count}_E + \sum \text{count}_D) \tag{4.8}$$

$$\text{planting\_area} \geq \frac{\text{tree\_width}}{4} \tag{4.9}$$

$$\sum \text{cost}_E + \sum \text{cost}_D \leq \$50,000 \tag{4.10}$$

$$\tag{4.11}$$

4.12: Constraints given by Choi and Lee [4] and implemented in iteration one Scenario One (maximising CO2 absorption).

To repurpose the existing code infrastructure to handle the Sundial Garden, a new class *EdinburghConstraints* was created. To maximise CO2 absorption, this fitness function worked by returning 0 if a constraint was violated or the total CO2 absorbed from the chromosome if all constraints were validated. The genetic algorithm is set up the same way as described in Chapter 3, with a 50% chance to apply the crossover function to a chromosome and a 20% chance to apply the mutate function to a chromosome.

The last changes come with a new greedy algorithm to initialise a valid starting grid. With no hedge or roadside planting constraints, no spots must be planted first. So, only one loop through the grid is needed. There is a $\frac{3}{1000}$ chance to plant a tree in any one plantable cell. If the chance is hit, a tree is chosen to help satisfy the current constraint violated. Once all constraints are satisfied, the grid is returned as a valid starting chromosome.

To initialise the *Grid* class, the Sundial Park was coloured black and white, where black represents a plantable area and white is an unplantable area. The result of this is shown in the Appendix Figure A.16. This is then resized so that 10 pixels represent one meter using Paint 3D [15]. Then, the picture is grayscaled and translated into a grid of the grayscale pixel values. Again, a 5x5 block of pixels is condensed into one cell so

that each cell now represents a half-by-half meter squared area. Lastly, to verify the creation of the grayscaled grid is accurate, random coordinates are selected from the original image shown in Figure 4.1 using Paint 3D, translated by hand, and checked on the new grayscaled grid. The verification is shown in the Appendix Figure A.17. The red ellipses location accurately matches the location of the coordinates chosen from 4.1, meaning the grayscaled grid is ready to be initialised into the *Grid* class.

Initialising the *Grid* class is much simpler in this scenario, where if the cell has a value of 255 (white), then the cell is unplantable. Otherwise, it is a plantable cell.

4.1.1.1 Maximising CO2 Absorption Results and Discussion

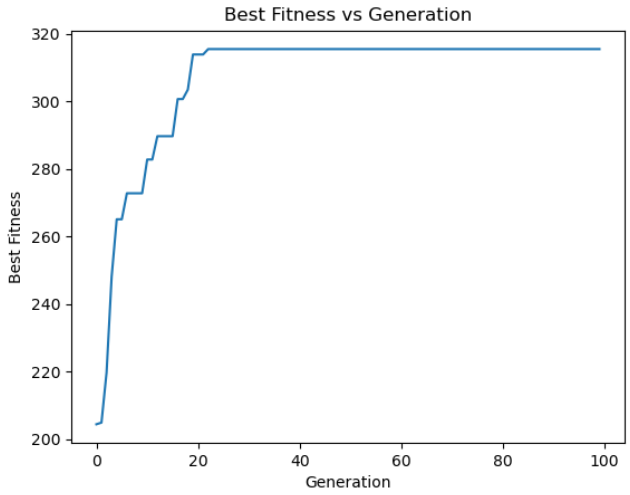

Figure 4.3: Best fitness of CO2 absorption for 500 population and 50 generations.

To run the algorithm, a 500 population sample is used and run for 50 generations. The average and best fitness scores across the generations are shown in the Appendix Figure A.15 and Figure 4.3. In both, it is clear that the algorithm quickly converges to an optimised layout. Given how small the planting area of the Sundial Garden is compared to the apartment complex, this is unsurprising as there are far fewer planting strategies to explore.

Overall, 45 trees were planted, with 16 of them being the Zelkova serrata, a continually popular tree in all scenarios and iterations. A total CO2 absorption of 315.5 kg/year was achieved at a cost of $43,649. The planting layout is shown in Figure 4.4.

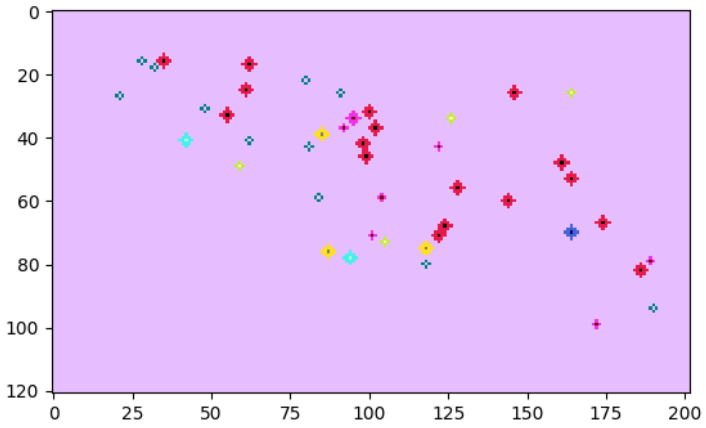

Figure 4.4: Optimised planting strategy for the Sundial Garden.

4.1.2 LiDAR Scanning

Shifting gears, the Sundial Garden needs to be viewed in the virtual and mixed reality application. Therefore, a 3D object of the area must be captured. To do this, a LiDAR scan of the Sundial Garden was taken using the Leica BLK350 G2, a LiDAR high-quality scanner available for load by the University of Edinburgh ucreate team. This scanner offers highly precise 3D capturing, which is necessary for visualising the planting layout shown in Figure 4.4 most accurately.

The LiDAR scanner, shown in the Appendix Figure A.18 generates a point cloud of the area. An example of this is shown in the Appendix Figure A.19. In total, 76 scans were taken of Sundial Garden; each scan is bundled together, as shown in the Appendix Figure A.20. Lastly, the point cloud is exported, and a mesh is created using the Cyclone 3DR application. The mesh creates triangles between the points, and the differing qualities of meshes simply change the number of triangles created. The more triangles, the smoother the resulting mesh looks.

4.2 Unity

Unity is a popular choice of game engine for creating virtual and mixed reality applications [13]. Furthermore, there is extensive documentation to creating these applications [23]. Therefore, it is a reasonable choice of game engine to create the planting strategy

visualisation in virtual and mixed reality.

The Varjo plugin is used within Unity to build a virtual and mixed-reality application with the Varjo XR-3. This plugin comes with integrated eye-tracking packages; however, the Leap package was additionally installed and used for hand tracking.

SteamVR base stations were used to track the headset. Although the Varjo XR-3 offers a beta headset tracking system without the need for base stations [2], it does not currently operate to the same standard provided by the SteamVR base stations.

4.2.1 3D Models

To best test the limitations of the Varjo XR-3 resolution, high-quality 3D objects were utilised from the Unity Store. For the trees, a package by Vvizart Studios is used [25]. Additionally, park benches from the Unity store were used to fill in the environment of the Sundial Garden and improve realism [24]. Lastly, a 2D texture is used to create the ground grass texture [1].

4.3 Unity Scenes

4.3.1 Virtual Reality

The application had two virtual reality scenes, one as a tutorial and the other as the Sundial Garden. Creating the tutorial scene visually consists of setting each tree type on a flat plane. The flat plane has a 2D grass texture. Lastly, a large wall is in front of the user's starting location, explaining the basic character functionality described in section 4.4.

The Sundial Garden went through two development phases. In the first phase, the ground from the LiDAR scan was included. However, the resulting 3D object ground visually was off, with the mesh triangles being completely unrealistic. In the Appendix, Figure A.21 depicts the ground before removing it. Additionally, the sundial shape was contorted and disrupted the immersive visuals. The ground was removed entirely to fix this, and a flat plane with the 2D grass texture was overlaid instead. Furthermore, a test between the medium-quality mesh and high-quality mesh is shown in the Appendix Figures A.22 and A.23. Visually, there is little difference between medium-quality and high-quality mesh. However, while the medium-quality mesh with no ground was 21 gigabytes, the high-quality mesh with no ground was over 40 gigabytes. Therefore, the minuscule improvements in the quality were not worth the performance-related issues

of a large object. Thus, the second phase implemented the medium-quality mesh with no ground.

With the border and ground in place, the trees, as given by the planting strategy in Figure 4.4, are then placed into the scene. Lastly, park benches are inputted throughout the garden to help with the immersive experience.

Figure 4.5 shows an example of the finalised virtual reality Sundial Garden in unity and seen through the Varjo XR-3 headset.

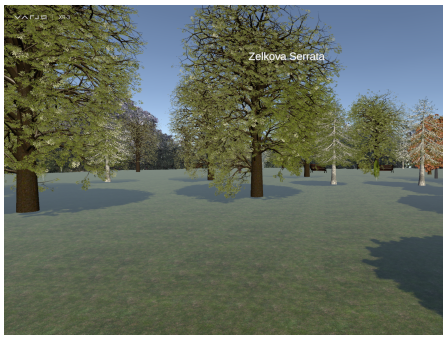

Figure 4.5: Virtual reality mode of the Sundial Garden.

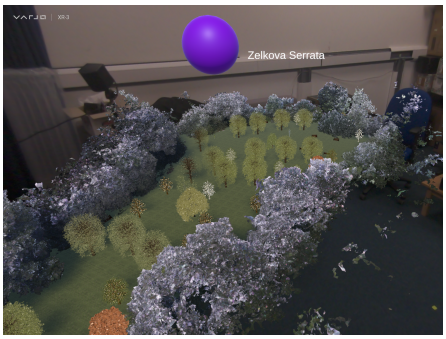

Figure 4.6: Mixed reality mode of the Sundial Garden.

4.3.2 Mixed Reality

Producing the mixed-reality scene is very straightforward with the Varjo XR-3 headset. To do this, the clear flags variable in the Unity camera object is changed from ”Skybox” to ”Solid Colour”. Then, the background RGB is set to 0 for each. This setting enables the pass-through. With this, the tutorial, virtual garden, and mixed-reality garden are set up visually. The final mixed reality Sundial garden is shown in Figure 4.6

4.4 Character Functionality

4.4.1 Manipulating Tree Position

One of the most important functionalities of this application is the ability to manipulate tree positioning. While the genetic algorithm gives an initial optimised layout, an urban planner may wish to modify it to some degree. Therefore, adding the functionality to move the trees is an essential feature.

This functionality puts to use the Leap library and Varjo XR-3 hand tracking. To move a tree, the user must grasp their right hand, which sends a raycast outwards from

the origin of their right hand. If this raycast strikes a movable object, such as a tree, then the tree is locked on to be moved. The radius between the user and the tree is the radius at which the tree moves around circularly. To move the tree closer, the user must take their closed fist and point it upwards, shortening the radius between the tree and the hand pulling it closer. To move the tree farther away, lengthen the radius by pointing the fist parallel to the ground again. As shown in the Appendix, Figures A.24 and A.25 showcase the positioning of the closed right fist in manipulating the tree radius.

Figure 4.7 shows moving the tree with the right hand. While fully functional, a major issue is that the raycast is invisible to the user. This makes it challenging to know what tree is being hit, if any. Therefore, an additional visual red laser is placed in the same direction the raycast is, allowing the user visual guidance on where the raycast is heading. This change is shown in Figure 4.8.

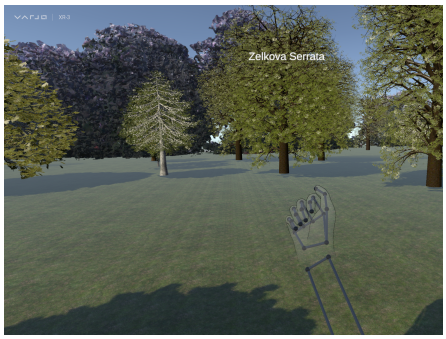

Figure 4.7: Moving the tree with no laser for guidance.

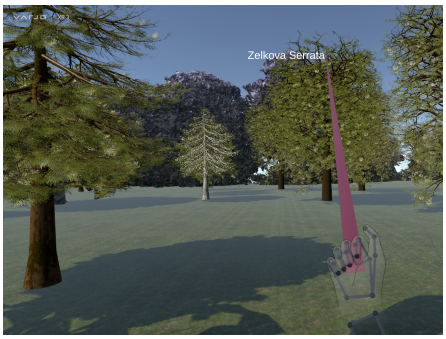

Figure 4.8: Moving the tree with red laser for guidance.

This functionality also works in the mixed reality mode, although it is less relevant. In this mode, using the laser results more often in hitting one tree when the user meant to move another. Given the smaller scale, it would be far more realistic to pick and move the trees like game board pieces. While the laser is the preferable option in virtual reality, it loses its applicability in mixed reality.

4.4.2 Character Movement

Moving the character in virtual reality currently lacks research [9], meaning there is no consensus in the literature for best practices. In this project, there were two distinct possibilities for movement. In early iterations, the character movement was enabled by keyboard controls. At the time, this was reasoned by the fact that since the headset itself was tethered to the desktop [2], then general movement by walking around was less

relevant. However, using a keyboard to move, while the right hand has integrated hand tracking proved to be a cumbersome experience. Furthermore, it completely detracts from the goal of creating an application without the burden of controllers.

To rectify, the final iteration of character movement again takes advantage of the hand tracking capabilities. The left hand now dictates movement, where the user will move in whichever direction they are currently looking when their left hand is closed. This choice is inspired by one of the potential solutions in the literature which suggest allowing players to fly around the environment [9].

In mixed reality mode, character movement is made redundant as the user can physically walk around the model-sized Sundial Garden.

4.4.3 Eye Tracking

The eye tracking is one of the main features the Varjo XR-3 advertises [2]. To showcase this functionality, the Varjo plugin was used to track the users gaze and display the tree specie's name they are currently looking at. The name is displayed as text to the user, which helps add in additional data visualisation of knowing which tree is planted in that location. One of the main issues with visualising the planting strategy in a 2D grid, as shown in Figure 4.4, is it is very difficult to understand which tree type is planted where. The eye tracking functionality rectifies this, allowing the user to finally visualise the planting strategy most accurately. Figures 4.7 and 4.8 show the eye tracking displaying the Zelkova serrata tree, which the user happens to be looking at.

4.4.4 Reset, Change, and Quit Scene

Lastly, three spheres are integrated into scenes: Reset, Change, and Quit Scene. True to their name, if the user uses their red laser to hit the sphere, the sphere will reset the changes made in the scene, change to the next scene, or quit the application. The change scene is what moves the character from Tutorial to virtual Sundial Garden. The sphere in virtual Sundial Garden initiates the mixed reality Sundial Garden, and the sphere here subsequently takes the user back to virtual Sundial Garden. It is these spheres that finalise the functionality and connect all the scenes into one application usable entirely without a controller.

Chapter 5

Conclusion

The genetic algorithm proved to be an effective optimisation algorithm for strategically planting trees with maximum CO2 absorption. This is all done while abiding by several important constraints used in Urban planning, such as hedge zones. While it failed to minimise cost as effectively, future work should experiment further with parameter tuning of the number of generations, population sizes, and how often the mutate and mate functions are applied to a chromosome. Furthermore, changing the greedy algorithm to initialise different starting chromosomes than scenario one may help the algorithm explore more and minimise more effectively. Regardless, the goal of maximising CO2 absorption was met.

Despite the algorithm's success, it proved to be a difficult task to comprehend the optimised planting strategy on a 2D graph. The Varjo XR-3, with its integrated hand and eye tracking, proved to be a useful tool to visualise complex data, such as this planting strategy effectively. It is also a valuable tool for urban planners to manipulate the optimised layout, ensuring the visual look is agreed upon before using the planting strategy. A significant limitation of the application is in the mixed reality mode, where a user cannot pick a tree with their hand to move it. While the laser is useful in virtual reality, it struggles to be effective in mixed reality. Future work should rectify this issue, allowing for a more natural functionality of moving trees like game board pieces to manipulate the planting layout. Furthermore, future work should implement an alert when manipulating the planting strategy and its effect on the constraints. For example, moving two trees too close to each other should tell the user that the planting radius is insufficient. In general, the functionality of the user and how it can manipulate the layout, such as deleting trees, adding in trees, and more, should be explored to create a product-ready application. Lastly, additional research on character movement

is necessary for virtual reality. While flying works, moving the head to face directly where the user wants to move can be tedious. This is especially noticeable when flying upwards and downwards. A greater consensus on best practices for virtual reality movement is highly relevant for this application.

Overall, this process can mostly be abstracted as an entire project to create a workflow relevant to more areas than just the Sundial Garden or apartment complex in Seoul, South Korea. With the introduction of Google Maps's API to create 3D objects anywhere in the world [18], it is becoming more accessible and easier to create virtual and mixed reality scenes without the need to LiDAR scan an area. A workflow can be integrated to take a coloured map, transform it into the correct dimensions, and output its grid of greyscaled values. This can easily flow into the genetic algorithm, which has all it needs to initialise the *Grid* class specific to the current space of interest. In Unity, the coordinates of the resulting algorithm can be coded to insert the trees into the Unity scenes computationally.

The only steps that need to be manually done are to create the Unity project and attach a script to handle the planting of trees in a scene. Additionally, someone would have to specify what the plantable area is and the colour of the image as well. However, these are relatively simple tasks, and most of this project can now be abstracted into a simple workflow to incorporate different areas. This project could be expanded with more time to include different sets of constraints, different trees, and output multiple grids to Unity to view. While the genetic algorithm is computationally expensive, especially for large planting areas, most of this project can be done computationally without requiring much manual work, making it a highly effective product.

Bibliography

[1] Terrain textures pack free — 2d nature — unity asset store. *assetstore.unity.com*. Accessed: Aug. 14, 2024.

[2] Varjo xr-3 - the industry's highest resolution mixed reality headset — varjo. *Varjo.com*. Accessed: Aug. 12, 2024.

[3] M. Asghari, A. M. Fathollahi-Fard, S. M. J. Mirzapour Al e hashem, and M. A. Dulebenets. Transformation and linearization techniques in optimization: A state-of-the-art survey. *Mathematics*, 10(2):283, 2022.

[4] J. Choi and G. Lee. Optimization of tree planting for urban residential green spaces. *Landscape Ecology and Engineering*, 19:107–121, 2023. Published: 07 October 2022, Issue Date: January 2023.

[5] Kurtis Danyluk, Teoman Ulusoy, Wei Wei, and Wesley Willett. Touch and beyond: Comparing physical and virtual reality visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 28(4):1930–1940, 2022.

[6] H. Fargeon, F. Pimont, N. Martin-StPaul, et al. Projections of fire danger under climate change over france: where do the greatest uncertainties lie? *Climatic Change*, 160:479–493, 2020.

[7] Indu Nashier Gahlawat and Poonam Lakra. Global climate change and its effects. *Integrated Journal of Social Sciences*, 7(1), 2020.

[8] M. Gen and L. Lin. Genetic algorithms and their applications. In H. Pham, editor, *Springer Handbook of Engineering Statistics*, Springer Handbooks. Springer, London, 2023.

[9] Devamardeep Hayatpur, Haijun Xia, and Daniel Wigdor. Datahop: Spatial data exploration in virtual reality. In *Proceedings of the 33rd Annual ACM Symposium*

on User Interface Software and Technology, UIST '20, page 818–828, New York, NY, USA, 2020. Association for Computing Machinery.

[10] Loren P. Hopkins, Deborah J. January-Bevers, Erin K. Caton, and Laura A. Campos. A simple tree planting framework to improve climate, air pollution, health, and urban heat in vulnerable locations using non-traditional partners. *People, Plants, and Planet*, 2021. First published: 02 December 2021.

[11] S. Katoch, S. S. Chauhan, and V. Kumar. A review on genetic algorithm: Past, present, and future. *Multimed Tools Appl*, 80(5):8091–8126, Feb. 2021.

[12] Luke Kemp, Chi Xu, Joanna Depledge, and Timothy M. Lenton. Climate endgame: Exploring catastrophic climate change scenarios. *Proceedings of the National Academy of Sciences (PNAS)*, 119(34):e2108146119, Aug 2022. Accessed: Aug. 14, 2024.

[13] E. H. Korkut and E. Surer. Visualization in virtual reality: A systematic review. *Virtual Reality*, 27(2):1447–1480, Jun. 2023. Received Mar. 14, 2022; Accepted Jan. 9, 2023; Published Jan. 17, 2023.

[14] Jian Lin. Developing a composite indicator to prioritize tree planting and protection locations. *Science of The Total Environment*, 717:137269, 2020.

[15] Microsoft Corporation. Paint 3d - free download and install on windows — microsoft store. *Microsoft Apps*, 2019. Accessed: Jul. 14, 2024.

[16] M. J. Nieuwenhuijsen. Green infrastructure and health. *Annual Review of Public Health*, 42:317–328, 2021.

[17] Charity Nyelele and Charles N. Kroll. A multi-objective decision support framework to prioritize tree planting locations in urban areas. *Landscape and Urban Planning*, 214:104172, 2021.

[18] Google Maps Platform. Blog: Create immersive 3d map experiences with photorealistic 3d tiles. *Google Maps Platform*, 2023. Accessed: Aug. 16, 2024.

[19] RHS. Find advice & tips on garden & indoor plants — plant finder & selector / rhs gardening. *Rhs.org.uk*, 2019. Accessed: Aug. 12, 2024.

[20] D. Sahija. Critical review of mixed reality integration with medical devices for patient care. *International Journal for Innovative Research in Multidisciplinary Field*, 8(1):100, Jan. 2022.

[21] Daniel Steel, C. Tyler DesRoches, and Kian Mintz-Woo. Climate change and the threat to civilization. *Proceedings of the National Academy of Sciences*, 119(42):e2210525119, 2022.

[22] The Independent. Apple Vision Pro vs Meta Quest 3: Is it worth paying seven times more?, Feb. 2024. Accessed: Feb. 13, 2024.

[23] Unity. Create vr & mixed reality experiences in unity. *Unity*, 2024. Accessed: Aug. 14, 2024.

[24] @UnityAssetStore. Parkchair. *Unity Asset Store*, 2016. Accessed: Aug. 14, 2024.

[25] @UnityAssetStore. Realistic forest tree pack. *Unity Asset Store*, 2024. Accessed: Aug. 14, 2024.

[26] Zixuan Wang, Yuguo Li, Jiyun Song, Kai Wang, Jing Xie, Pak Wai Chan, Chao Ren, and Silvana Di Sabatino. Modelling and optimizing tree planning for urban climate in a subtropical high-density city. *Urban Climate*, 43, 2022.

[27] Kathleen L. Wolf, Sharon T. Lam, Jennifer K. McKeen, Gregory R.A. Richardson, Matilda van den Bosch, and Adrina C. Bardekjian. Urban trees and human health: A scoping review. *International Journal of Environmental Research and Public Health*, 17(12), 2020.

Appendix A

First appendix

A.1 Planting Optimisation

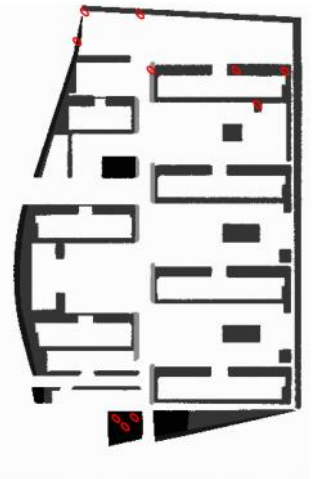

Figure A.1: Dimension check of the updated grid where each cell represents a half-by-half meter square area.

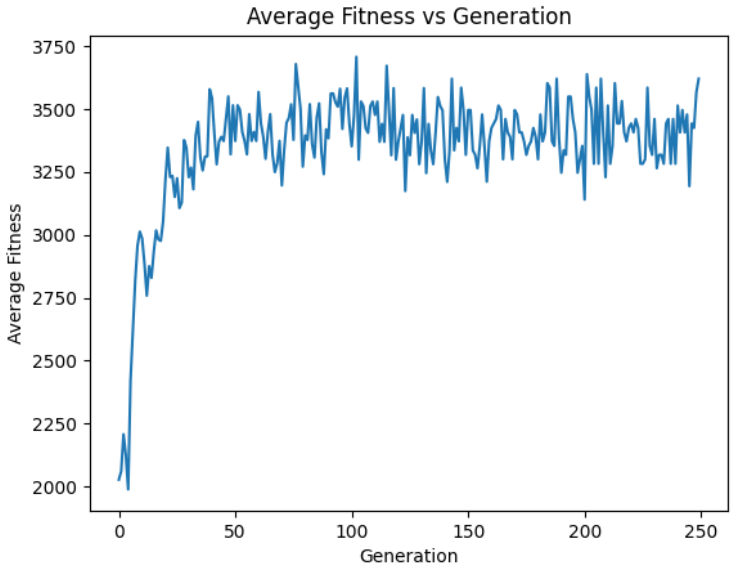

Figure A.2: Average fitness score for iteration one scenario one for a population size 250 and 250 generations.

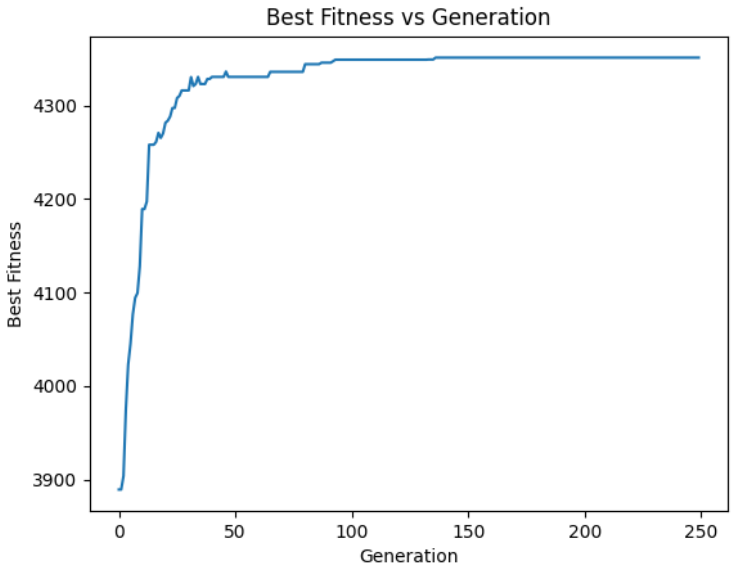

Figure A.3: Best fitness score for iteration one scenario one of a population size of 250 and 250 generations.

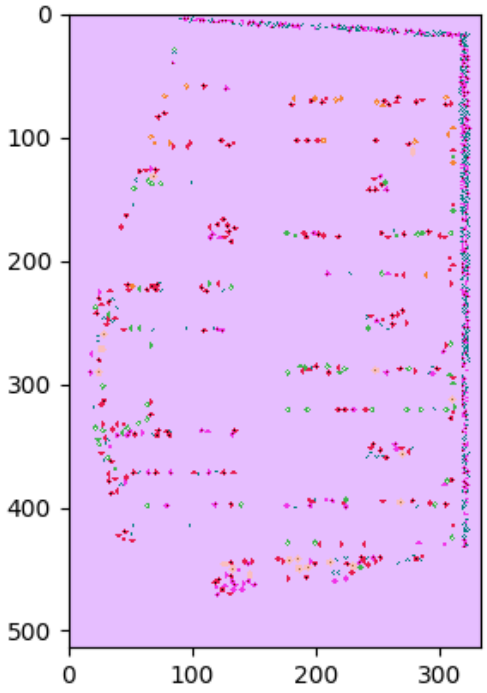

Figure A.4: Planting strategy for the best grid of 250 population size for iteration one scenario one.

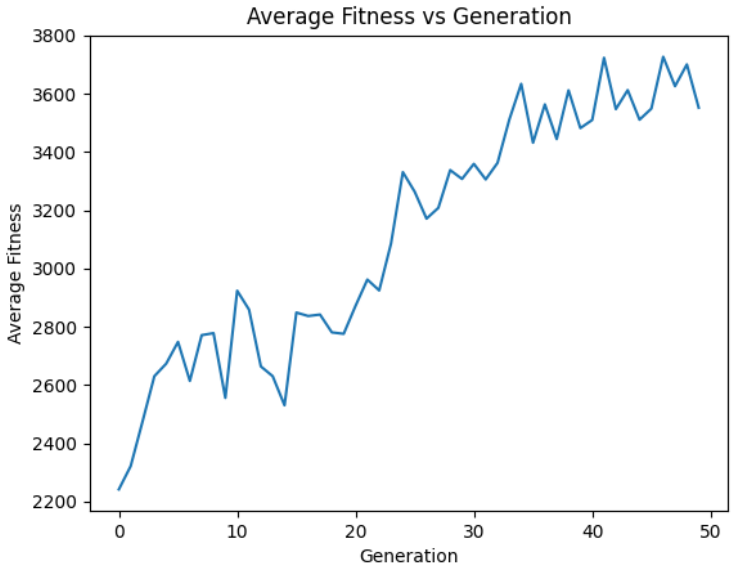

Figure A.5: Average fitness score for iteration two scenario one for a population size 500 and 50 generations.

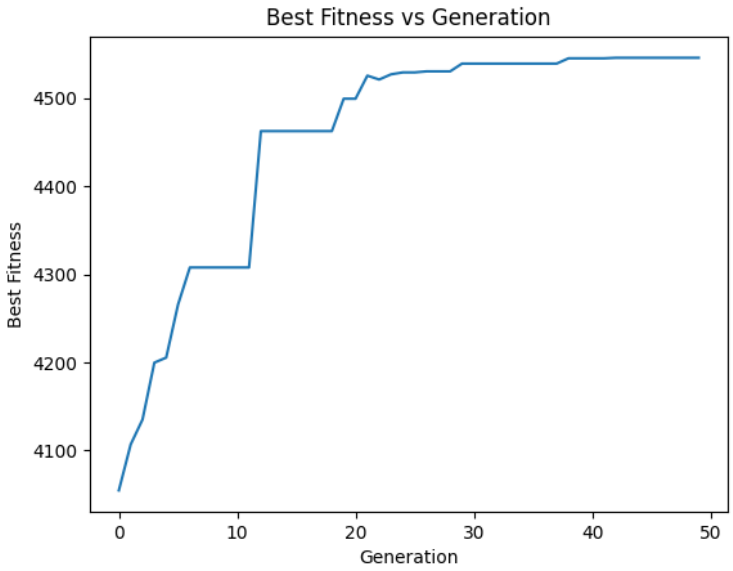

Figure A.6: Best fitness score for iteration two scenario one for a population size 500 and 50 generations.

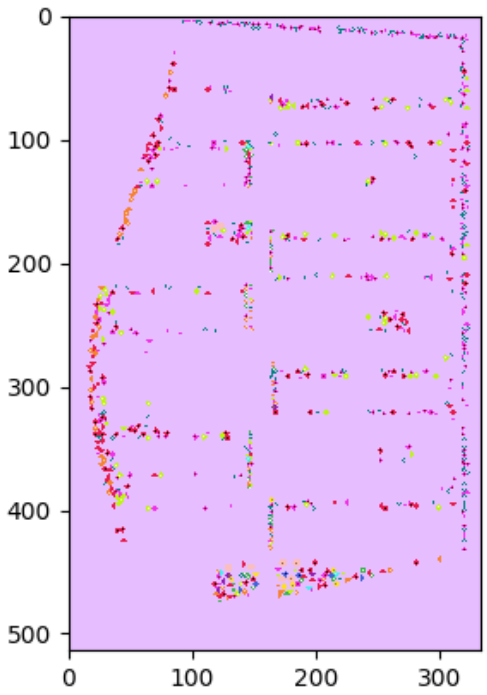

Figure A.7: Best layout from best fitness score for iteration two scenario one for a population size 500 and 50 generations.

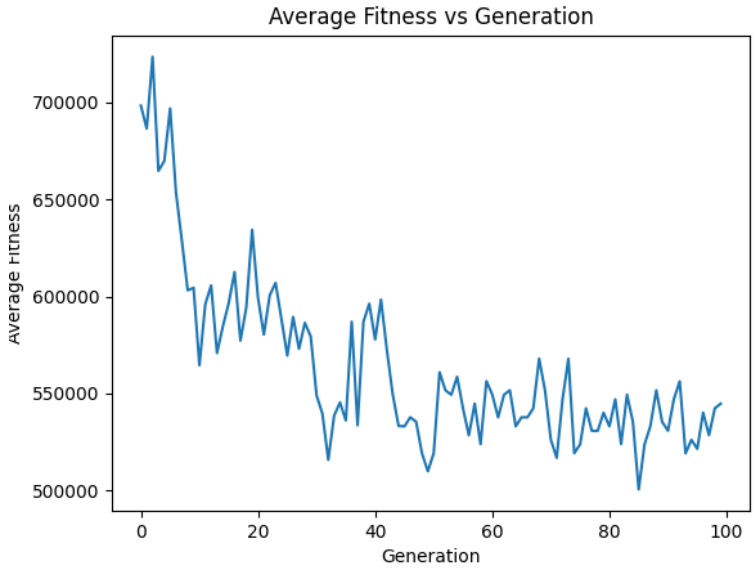

Figure A.8: Average fitness scores for iteration two scenario two for 250 population and 100 generations.

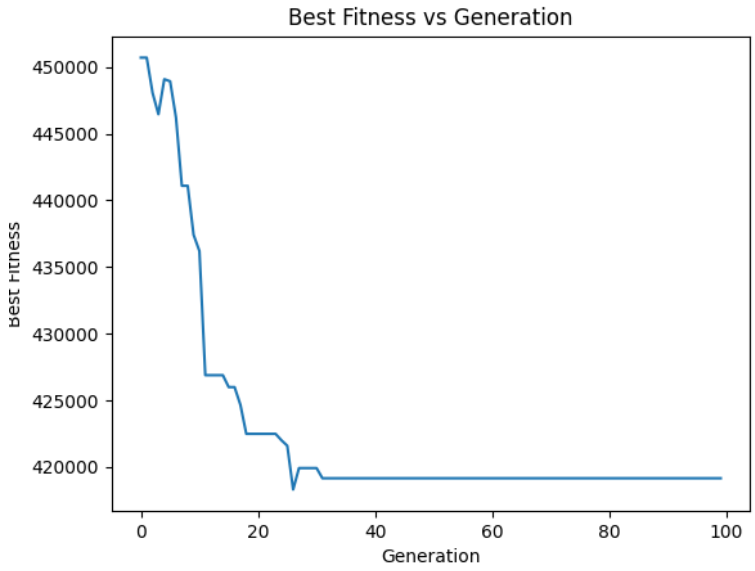

Figure A.9: Best fitness scores for iteration two scenario two for 250 population and 100 generations.

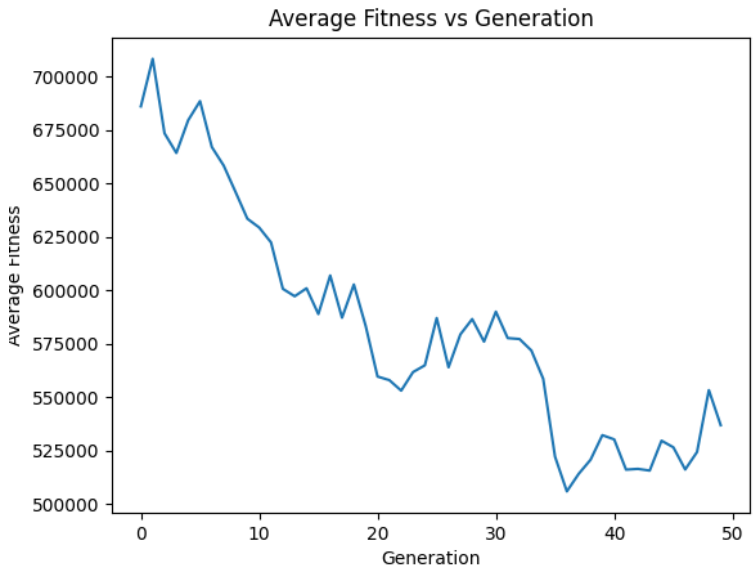

Figure A.10: Average fitness scores for iteration two scenario two for 500 population and 50 generations.

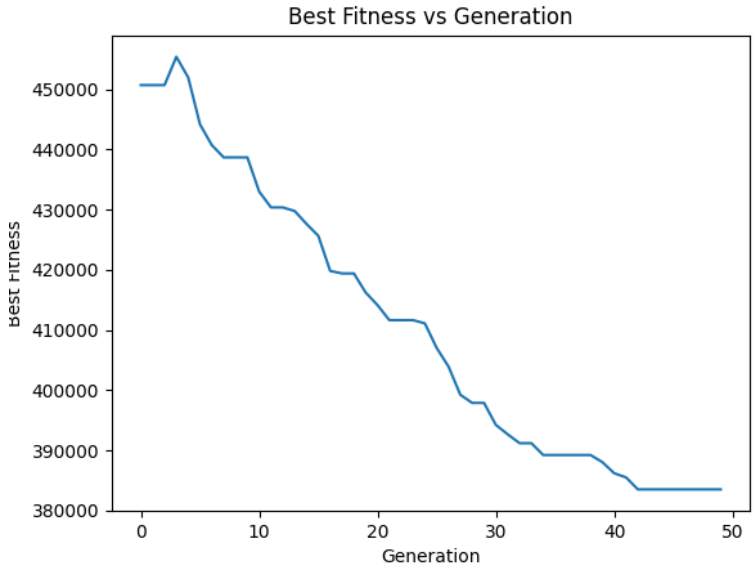

Figure A.11: Best fitness scores for scenario two for 500 population and 50 generations.

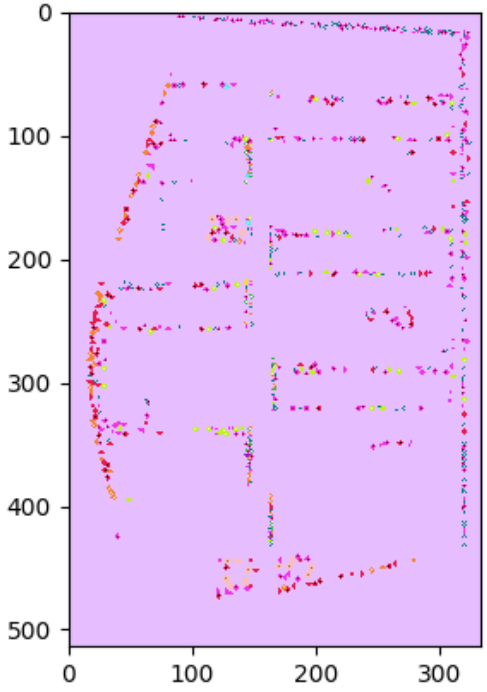

Figure A.12: Planting strategy for iteration two scenario two for 250 population and 100 generations.

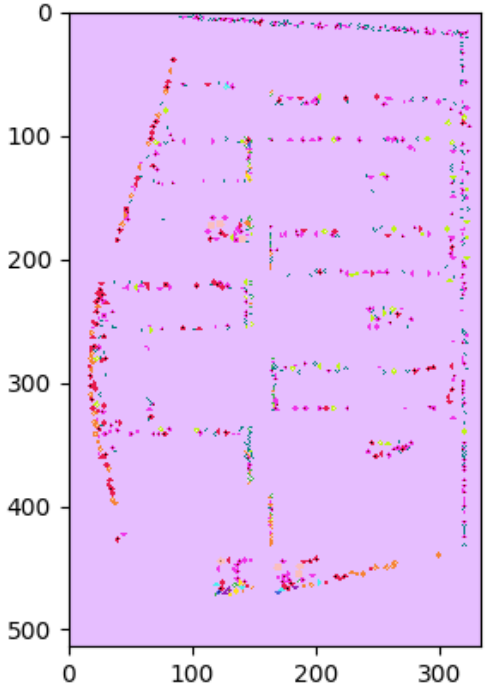

Figure A.13: Planting strategy for iteration two scenario two for 500 population and 50 generations.

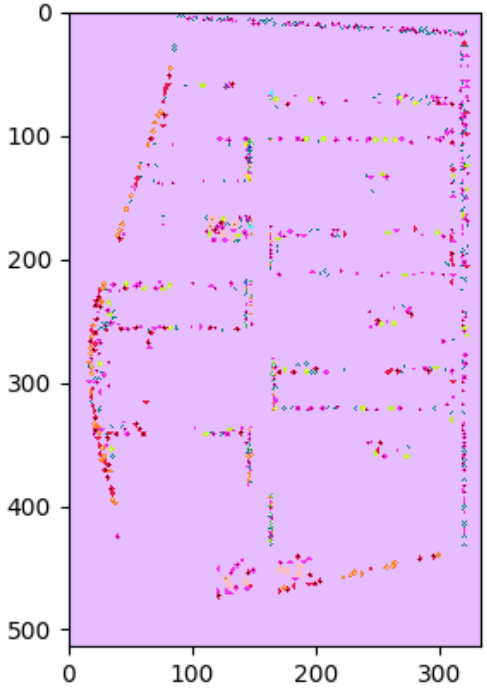

Figure A.14: Planting strategy for iteration two scenario two for 1000 population and 50 generations.

A.2 Virtual and Mixed Reality Data Visualisation

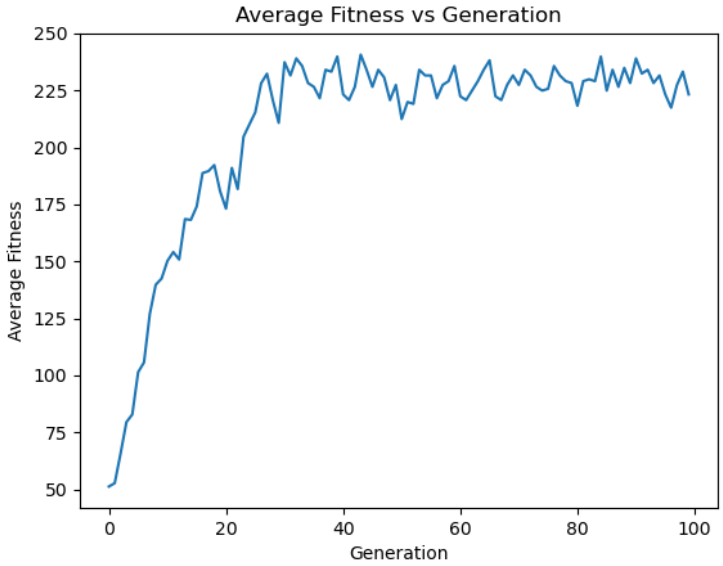

Figure A.15: Average fitness of CO2 absorption for 500 population and 50 generations.

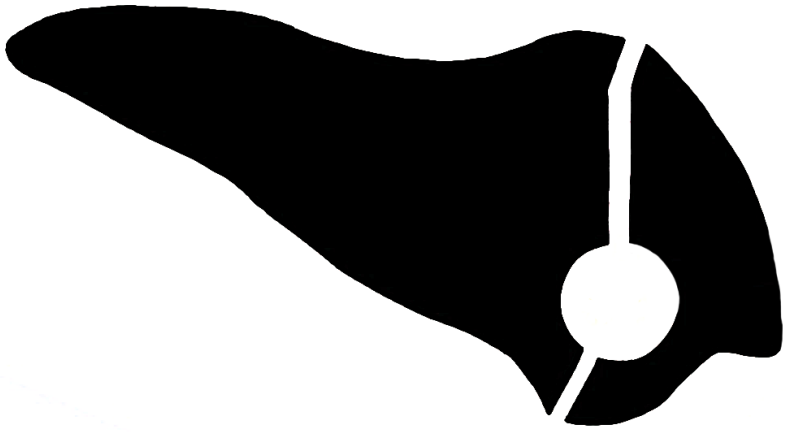

Figure A.16: Sundial Park coloured to prepare for grayscale.

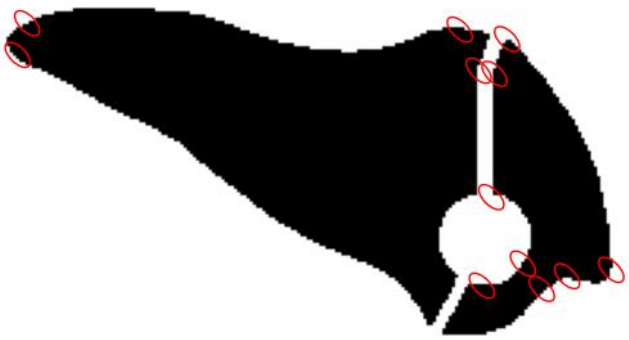

Figure A.17: Grayscale dimension checking.

Figure A.18: Leica G2 LiDAR scanner creating scan in Sundial Garden.

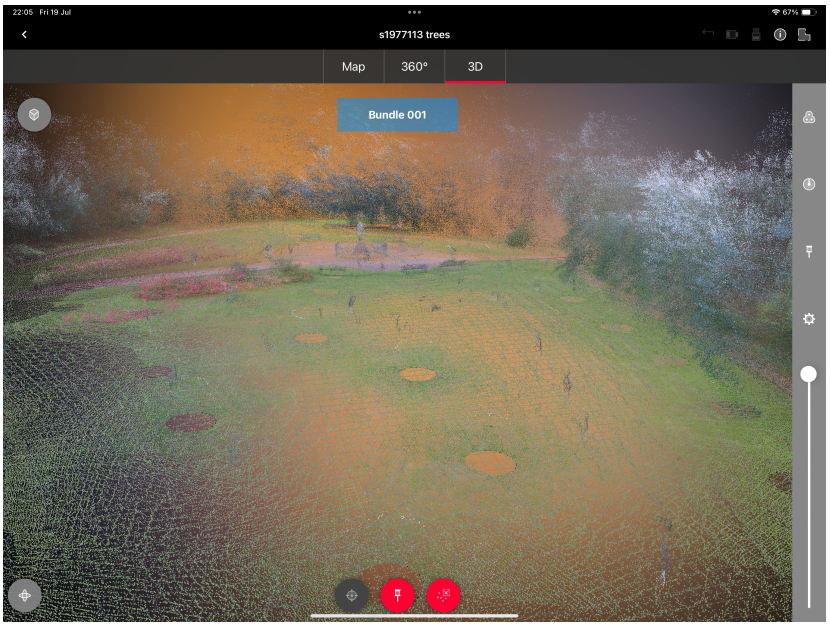

Figure A.19: Point cloud generated by the LiDAR scanner.

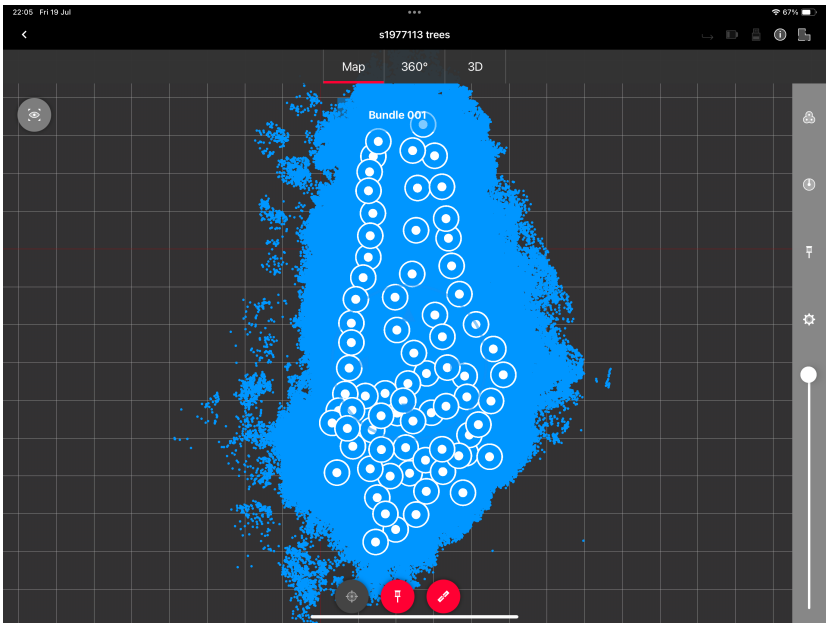

Figure A.20: The finalised bundle of 76 scans taken in Sundial Garden.

Figure A.21: 3D mesh with ground included viewed in Unity.

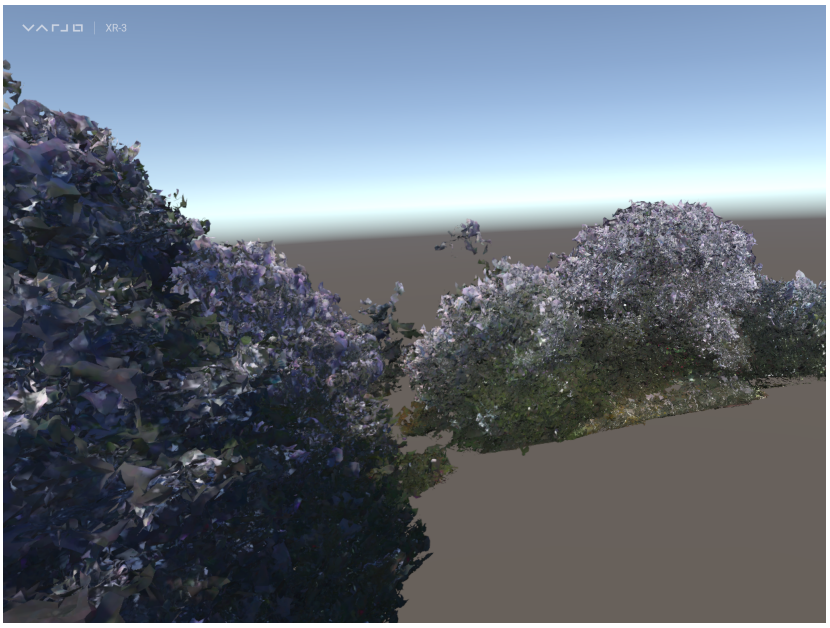

Figure A.22: Medium quality 3D mesh viewed in Unity.

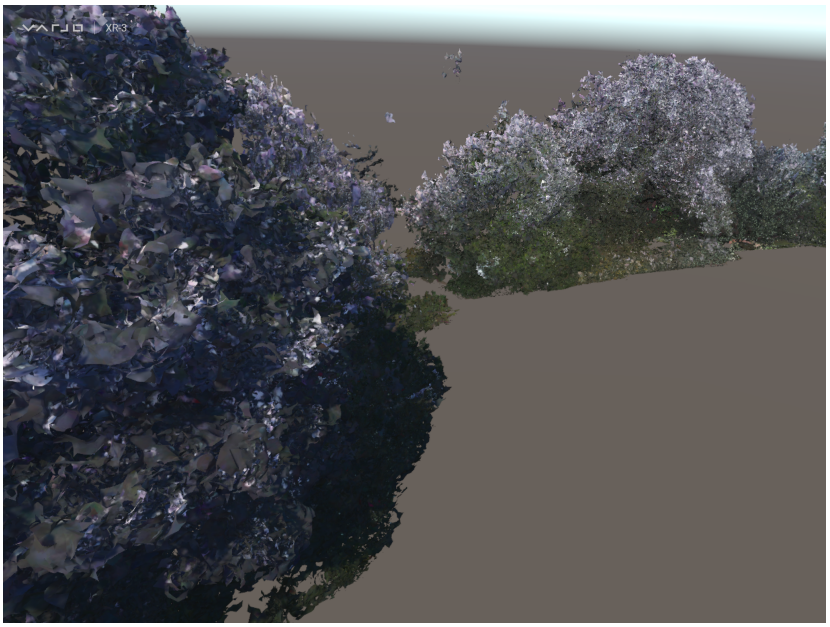

Figure A.23: High quality 3D mesh viewed in Unity.

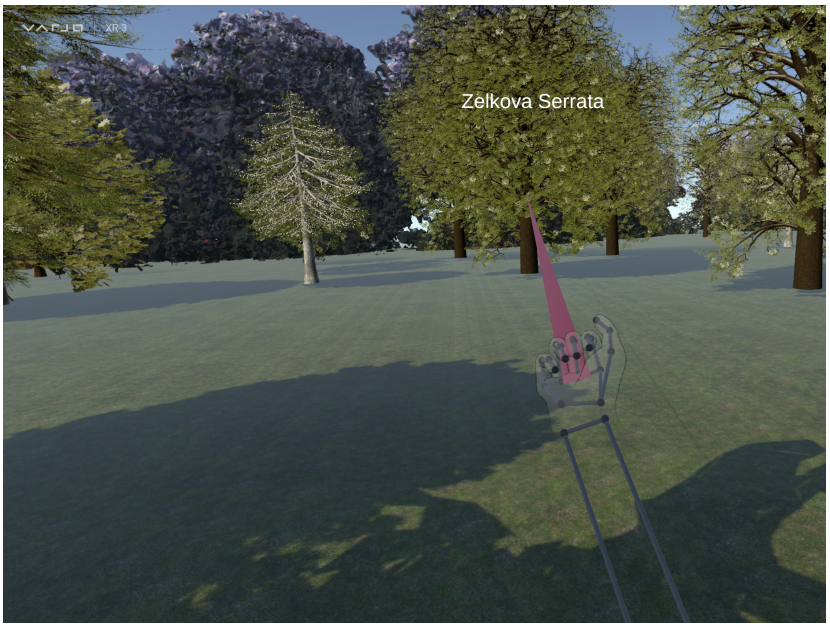

Figure A.24: Initial start of locking onto tree.

Figure A.25: Shortening radius by moving fist upwards, bringing tree closer to user.