Tree Planting Optimisation and its Applications with VR/MR Visualisation

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

1.2 Significance

1.3 Overview

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 [9]. This built greatly upon existing tree priority schemes, which tend to be environment-specific, whereas Lin's framework accounts for environmental variability [9]. Lin utilised the i-Tree Landscape tool to build a tree priority protection index (PPI) [9]. 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 [9]. The PPIs are constructed from three categories: environment, tree, and human. Then, these are compiled using the equation specified in equation (2.1) [9].

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

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 [9]. 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 [9]. 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 [11]. The framework is then optimised using linear programming [11]. 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 [11]. Each scenario maximises a different singular objective while adhering to the other objectives' constraints [11]. 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 [6]. 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 [6]. 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 [6]. Lastly, the health department then worked with the local environmental group to raise awareness about the project and its positive effects [6].

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 [6]. 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 [14]. 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 [14]. 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 [14]. 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 [2]. Using an apartment complex in Seoul, South Korea, as a case study, Choi and Lee employ a linear programming solution [2] 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 [2]. Two scenarios were optimised through these constraints: total tree CO2 absorption and species diversity, and one minimisation scenario for total cost [2].

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 [2]. 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 [7]. 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 [7]. 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 [7].

The genetic algorithm has variations, such as the binary-coded genetic algorithm. In this case, chromosomes are binary [7]. This variant, however, has two issues. Firstly, the method has a high computational complexity [7]. 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 [1].

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 $z_{ij} = x_i \times y_j, \forall i \in 1,...,m, \forall j \in 1,...,n$ [1]. Asghari also describes other functions such as the square root function, maximum and minimum, and multiplication of continuous variables similarly to linearise [1]. 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 [4]. Additionally, genetic algorithms are adaptable to any problem, whether linear or nonlinear and discrete or continuous [4]. 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 [4].

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 [4]. 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 [13]. Virtual reality's immersiveness has also been utilised for visual data analytics and visualisation [5].

A 2020 article introduced DataHop, a virtual reality data visualisation tool equipped with a filter to modify data [5]. 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 [5]. A controller was used to interact with the displays [5].

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 [5]. 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 [5]. While the users found the software helpful, there were concerns about the vastness of the environment, which could become disorienting [5]. 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 [5], and they self-rated their data visualisation expertise [5]. 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 [12]. 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 [12]. In these 30 papers and articles, 26% found the integration enhanced clinical decision-making for patient diagnosis and treatment [12]. 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 [12]. 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 [3]. 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 [3]. 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 [3]. In the virtual, there was no difference between the hand scale and table scale, but the room-scale led to considerably slower times [3].

The physical portion was set up similarly, using physical objects instead of the virtual. Eighteen total participants through a university were recruited [3]. 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 [3]. 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 [8]. 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 [8]. The different movement techniques previously mentioned by Danyluk et al. are mentioned, like jump, portal, and teleportation [8, 3]. To create the visualisations, Unity is overwhelmingly chosen, making up 37.9% of the total software [8]. Notably, issues with the current hardware are addressed, such as cybersickness, latency in tracking, and low refresh rates [8]. Cybersickness is when the user feels nauseated and can be partially caused by latency and refresh rates, which break the immersion [8]. 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 [2] 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 [2]. 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.

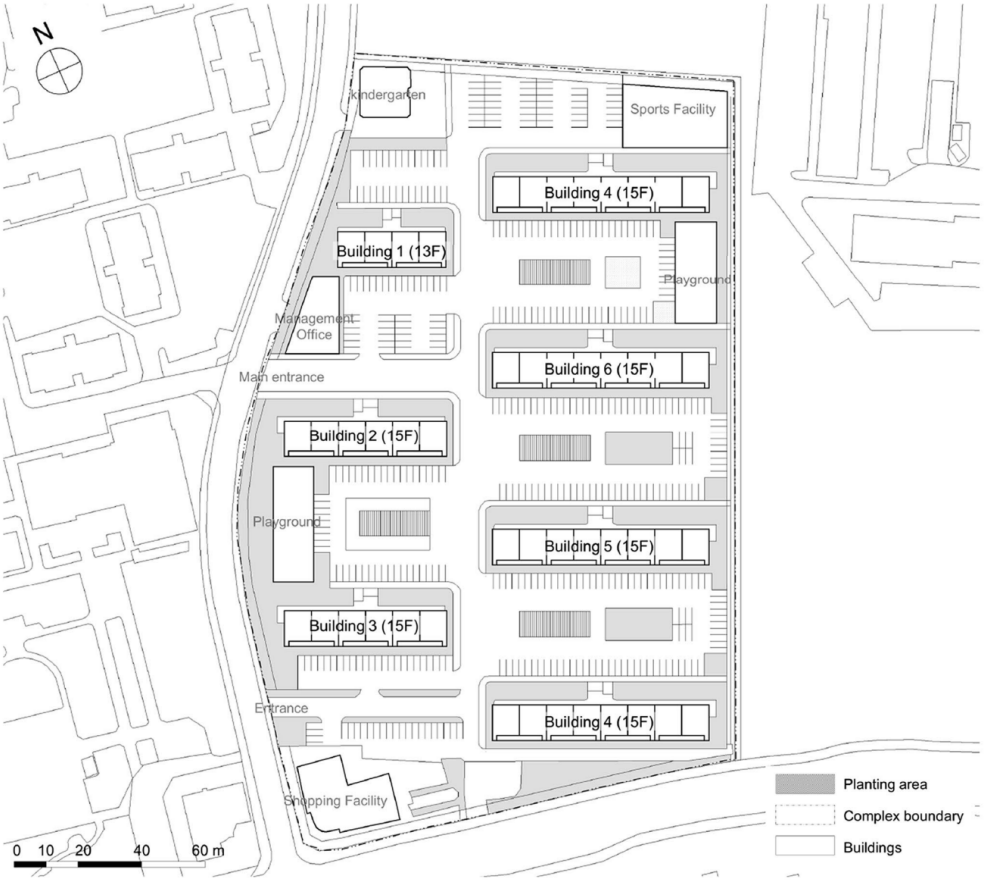

Figure 3.1: Apartment complex in Seoul, South Korea [2]

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 [2]. 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 [2]. 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 3.2. 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 [2]

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.

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

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

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

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

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

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

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

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

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

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

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

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

3.13: Constraints given by Choi and Lee [2] 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 [2].

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, 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 their classes for each scenario. However, there are only minor differences between these classes. The classes act 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 mutate function has a random chance of 5% to mutate. The tree selected is completely random. A random seed of 100 is set for reproducible results. There is a 50% chance to apply the crossover function and a 20% chance to apply the mutate function. 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.

Scenario two minimises cost and works similarly, although this scenario is only implemented in the second iteration. One difference is the weight of the fitness score set to -1.0 to minimise. The second 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.

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 [10]. 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.

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. Figure 3.4 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.

Using the values, the Square class with the correct attributes could be initialised

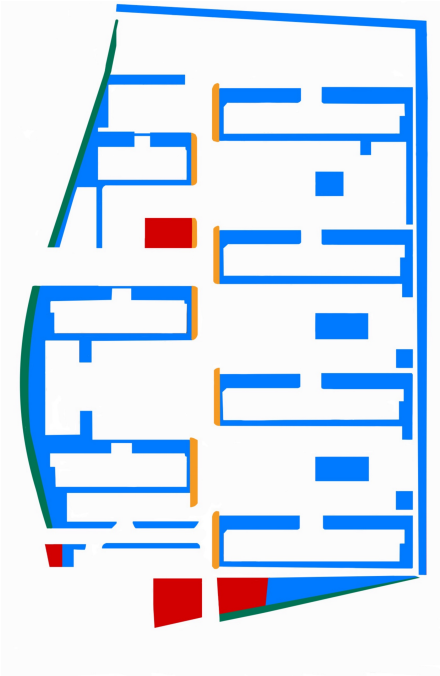

Figure 3.3: Apartment complex coloured. Blue represents plantable areas. Red is large tree planting, while green is roadside planting. Lastly, the orange depicts the pedestrian road planting.

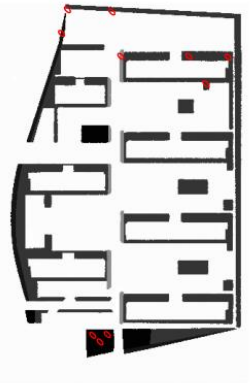

Figure 3.4: Dimension check of the updated grid where each cell represents a half-by-half meter square area.

Table 3.1: Colour To GrayScale Values

Colour	Grayscaled Value
Blue	101
Red	63
Green	77
Orange	168

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 3.5 and 3.6 compare before and after filling in gaps.

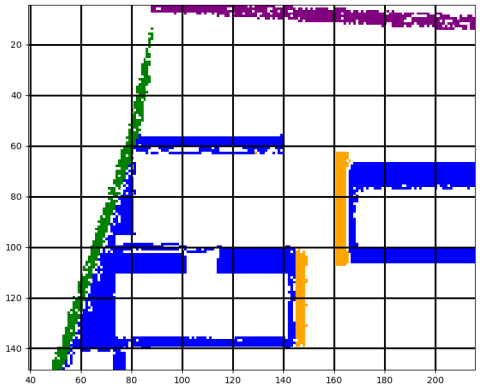

Figure 3.5: Grid class initialised before gaps filled in.

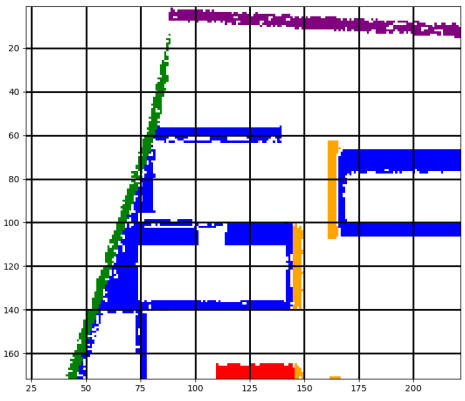

Figure 3.6: Grid class initialised after gaps filled in.

Although not perfect, many of the inconsistencies are now rectified. Furthermore, the decision was made to under-compensate 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.

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.

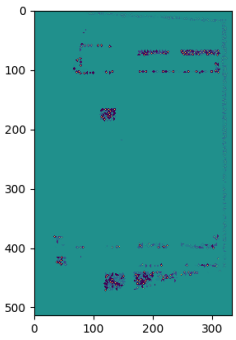

Figure 3.7: Valid starting state returned by greedy algorithm planting in each plantable spot.

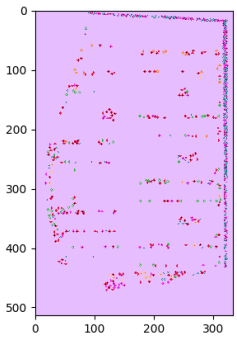

Figure 3.8: Valid starting state returned by greedy algorithm planting a specified percentage of the time in a plantable spot.

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. Therefore, a more aesthetic layout can be achieved by tweaking the greedy algorithm. 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 rest of the plantable area. These changes resulted in Figure 3.8, a much

more aesthetic starting chromosome and more relevant to optimising a planting strategy in an apartment complex.

3.1.2 Scenario One

3.1.2.1 Results

3.2 Iteration Two

added in last two constraints. added in scenario two (cost min.)

3.2.1 Adding in Remaining Constraints

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

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

[2]. 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 3.14 through 3.25. Note that equation 3.25 requires that a minimum CO2 absorption of all trees be greater than 3500 kg/year [2].

$$\sum \text{credits}_E + \sum \text{credits}_D \geq 0.2 \times A_L \tag{3.14}$$

$$\sum \text{credits}_E \geq 0.2 \times (\sum \text{credits}_E + \sum \text{credits}_D) \tag{3.15}$$

$$\sum \text{credits}_D \geq 0.2 \times (\sum \text{credits}_E + \sum \text{credits}_D) \tag{3.16}$$

$$\sum \text{credits}_{NE} + \sum \text{credits}_{ND} \geq 0.1 \times (\sum \text{credits}_E + \sum \text{credits}_D) \tag{3.17}$$

$$\sum \text{count}_{LE} + \sum \text{count}_{LD} \geq 0.06 \times (\sum \text{count}_E + \sum \text{count}_D) \tag{3.18}$$

$$\sum \text{canopy}_E + \sum \text{canopy}_D \leq 0.6 \times A_L \tag{3.19}$$

$$\sum \text{canopy}_E + \sum \text{canopy}_D \geq 0.4 \times A_L \tag{3.20}$$

$$\sum \text{count}_E \geq 0.015 \times (\sum \text{count}_E + \sum \text{count}_D) \tag{3.21}$$

$$\sum \text{count}_D \geq 0.015 \times (\sum \text{count}_E + \sum \text{count}_D) \tag{3.22}$$

$$\text{Length}_{SR} \leq \sum \text{crown\_width}_H \tag{3.23}$$

$$\text{planting\_area} \geq \frac{\text{tree\_width}}{4} \tag{3.24}$$

$$\sum \text{co2\_absorption}_E + \sum \text{co2\_absorption}_D \geq 3500 \tag{3.25}$$

3.26: Constraints given by Choi and Lee [2] and implemented in iteration one Scenario Two (minimising cost).

3.2.2 Genetic and Greedy Algorithm Updates

3.2.3 Scenario One

3.2.3.1 Results

also compare to iteration one

3.2.4 Scenario Two

3.2.4.1 Results

Chapter 4

Visualisation with Varjo XR-3

Insert purpose. goal of no controllers only hands. point of visualisation and manipulate the optimised layout and observe in both vr and mr.

4.1 Unity

insert libraries used. varjo plugin, etc.

4.1.1 3D Models

4.2 Edinburgh Inverleith Park

4.2.1 Repurposing Genetic and Greedy Algorithms

4.2.2 LiDar Scanning

4.3 Unity Scenes

4.3.1 Virtual Reality

* iteration 1: high quality ground * iteration 2: medium quality no ground compared to high quality no ground

4.3.2 Mixed Reality

4.4 Character Functionality

4.4.1 Manipulating Tree Position

* iteration 1: no laser * iteration 2: laser

4.4.2 Character Movement

* iteration 1: movement only given by length of tethered cord to computer. keyboard considered to improve this but defeats purpose of needing controller. * iteration 2: left hand to move around

4.4.3 Eye Tracking

4.4.4 Reset, Change, and Quit Scene

Chapter 5

Conclusion

Bibliography

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

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

[14] 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.

Appendix A

First appendix

A.1 First section

Any appendices, including any required ethics information, should be included after the references.

Markers do not have to consider appendices. Make sure that your contributions are made clear in the main body of the dissertation (within the page limit).

Appendix B

Participants' information sheet

If you had human participants, include key information that they were given in an appendix, and point to it from the ethics declaration.

Appendix C

Participants' consent form

If you had human participants, include information about how consent was gathered in an appendix, and point to it from the ethics declaration.