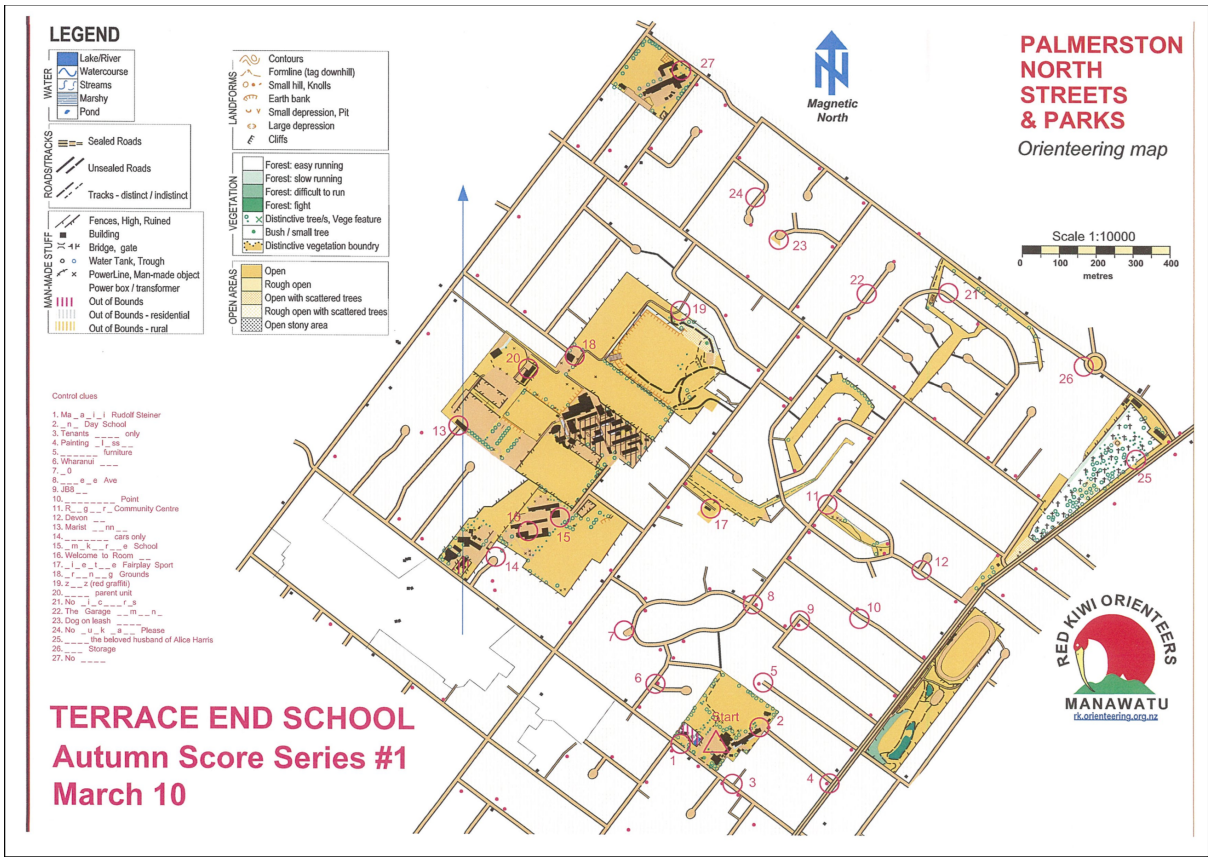
Assignment 2: Orienteering Optimisation
Simulation Modelling and Optimisation 282 758

Marc Alexander Sferrazza
12164165 *†



MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

*This work was not supported by any organization

†Faculty of Mechatronics Engineering, Massey University, Albany, Auckland, New Zealand Progress of project: https://github.com/alex1v1a/Simulation-Modelling-and-Optimisation/

Contents

List of Figures

1. INTRODUCTION

An orienteering course has a number of controls that can be reached. The task is to optimise the result for various running speeds on which a runner can attempt to complete the course, based on a certain points scheme. Using a branch of optimisation called evolutionary optimisation, the GAOT (genetic algorithm optimisation toolbox - 1995) is to be implemented, to take advantage of it's class, or group of optimisation techniques. The Biological evolution techniques are modelled in Matlab and results extracted to this report.

1.1. Overview

The orienteering course is shown below and the target points values are worth either 2, 3 or 4 points. There are 27 controls total, 9 of each worth 2, 3, and 4 points respectively. Runners have 60 minutes to maximise their score with a penalty of 2 points per 30 seconds late back.

The constraints and limitations indicate that a runner must remain on the yellow marked paths at all times during the simulation, to otherwise therefore not cross any private grounds, and/or walk through buildings etc.

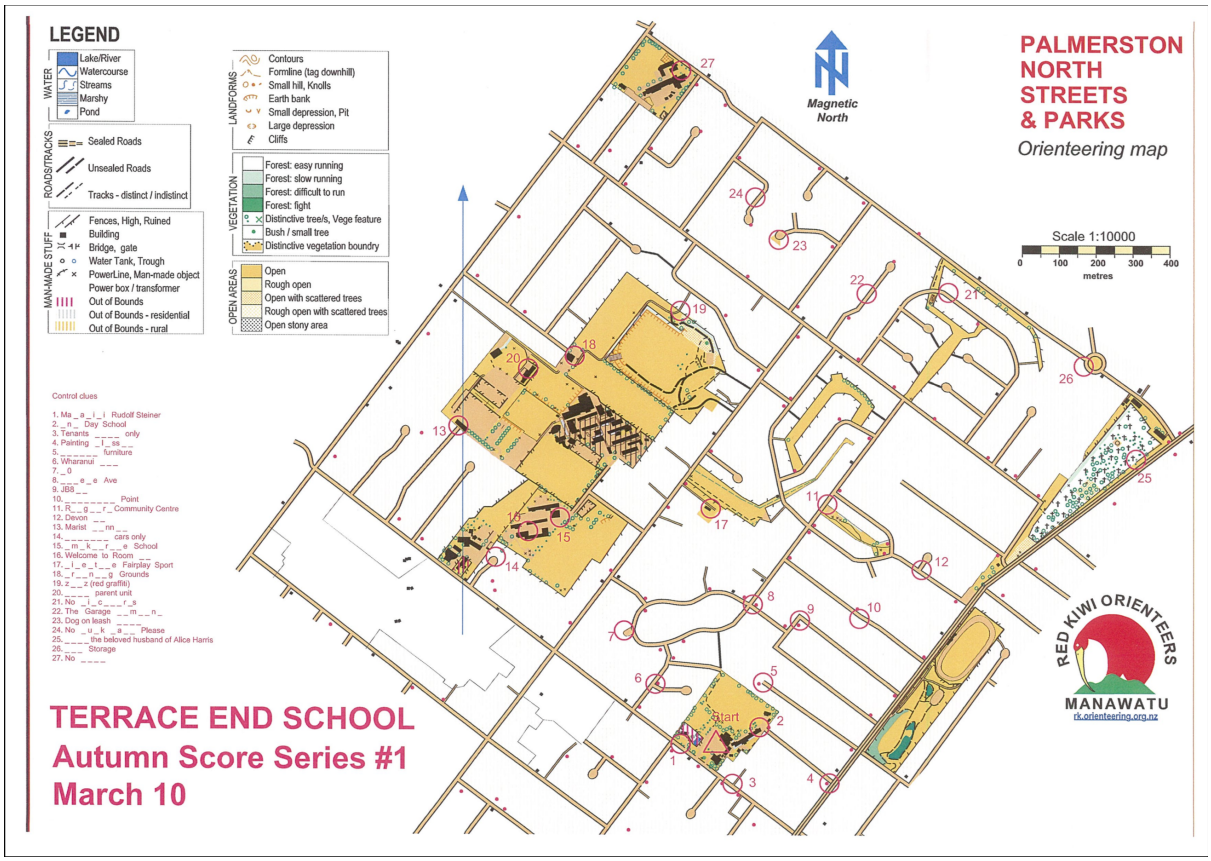


Figure 2: Terrace End School Points Course

2. METHOD

The Matlab model to be presented is based on Setup Code* ¿ Opt Function ¿ Objective function*

After locating the control points on the map and their respective distances to one another on a legal term based from the constraints; the points are then translated to an array matrix which can be used to easily lay all possible distances to each point in all combinations, of which 729 terms are found.

From the combinations of distances the Genetic Algorithm function can be used to calculate the optimal route for runner based on the given speed, with the shortest distance.

2.1. Genetic Algorithms

Using a Genetic Algorithm the method of natural selection (Darwinism - Survival of the fittest, or in this case best performance) can been deemed appropriate based on the biological evolution techniques described in the search algorithm. Therefore applying this technique to the 27 combinational control options the runner can achieve in 60 min, the maximum score can be calculated by simulation of exclusions to the weakest links.

Eventually the model will find the optimal solution based on the runners speed after shedding all other less efficient options, like that of the process of natural evolution systems. It does this based on a randomised search; however while the method is randomised the past results and background informations are used to find a region of interest with best performance, to narrow the scope of the search and find a optimal result in a faster given time.

2.2. Distance and Coordinates Allocation

Using Google maps the distances are recorded from each control point, traveling centred along each path and scaled accordingly. In order to set up the Genetic Algorithm the distances are then translated to a matrix with every possible distance from each control device mapped (27x27 array of 729 terms) - The matrix then is converted to the shortest distances between controls.

2.3. Tracker Function

Please note the orderBasedExample has been used with tspEval (modified to tracker) from the GAOT toolbox for this section.

To find the shortest route for the runners the Genetic Algorithm function needs to be called, but scoring for each point must be taken into consideration and therefore implemented with the Genetic Algorithm function. (Please see fig. orderBasedExample)

The adaptive aspect of the function loops and gets results for every iteration in the target, scores are added and the acquired distance based on a rotated location is incremented on the common term. The input passed is the runners speed; other inputs and outputs are also available like the obvious total points or final score. The scores are added - there are 9 of each, 2, 3 and 4 - when the runner reaches each control formed on a coordinate list. The cycle continues until the position doesn't change then exits, where the position is overwritten and run again until the loop is complete.

The penalties are then deducted from the total score after the 60 min time limit is reached, a 2 point reduction is taken per 30 seconds the runner returns late. (Please refer to fig. tracker with commentary for further details)

A full code snippet has been provided in the appendix for full step by step details please review comments.

2.4. Optimisation Process

3. RESULTS

4. CONCLUSIONS

For any progress related to the report please see the public Github repo for alex1v1a or use the link in the cover page to be automatically redirected to this project. The repo provides all relative project information

I have come to a further understanding learned the fundamentals of ROS with the use of such simulation tools like Gazebo. I am aware of other simulation tools available but due to the community support this is a great place to start. The use of plugins with models to be controlled by the teleop python script is used in calibration with the rviz graphical output to help visualise the robots (models) sight.

This is a powerful tool and from the basic level of understanding is an extremely vital aspect of the mechatronics processing, it is a great skill to learn.

APPENDIX

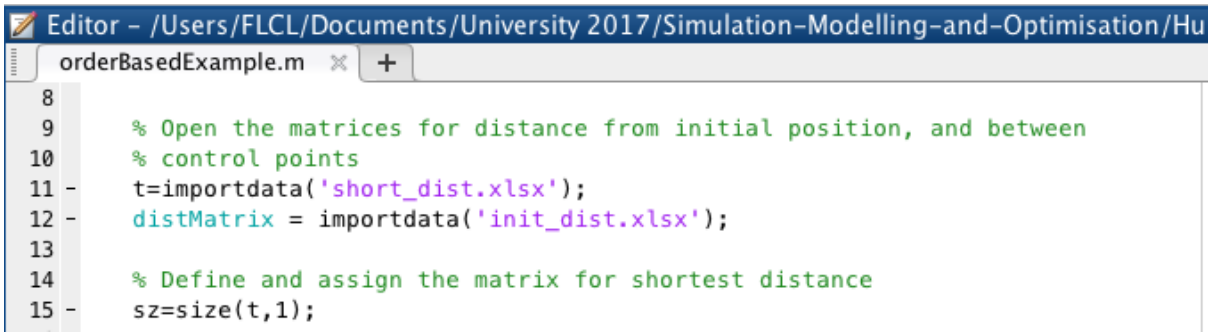


Figure 3: Import the dist matrices to orderBasedExample

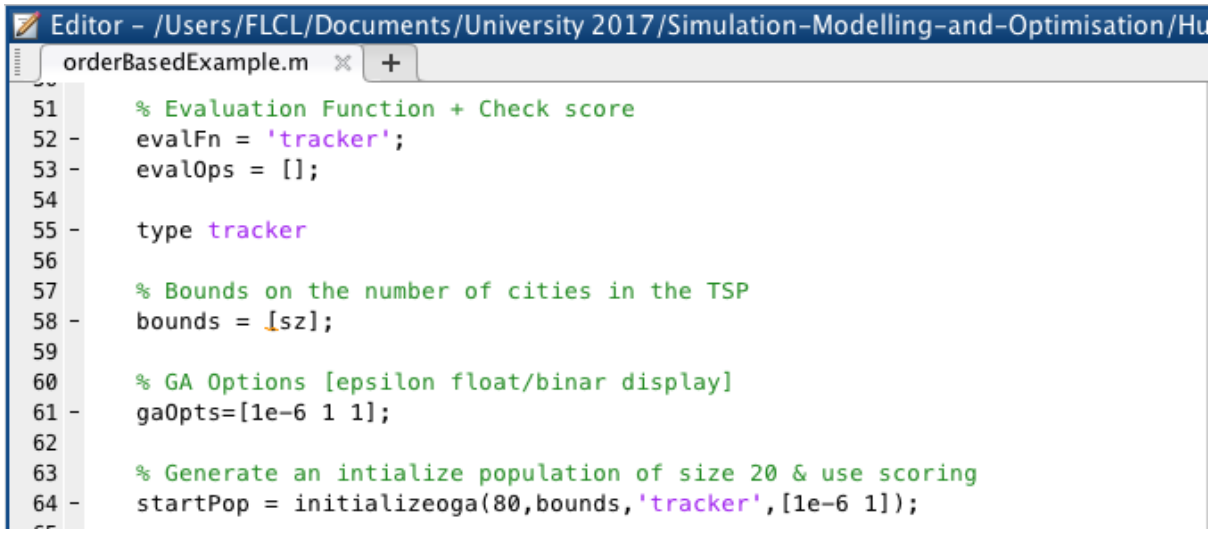


Figure 4: Preform Genetic Algorithm and call function ”tracker” for scoring

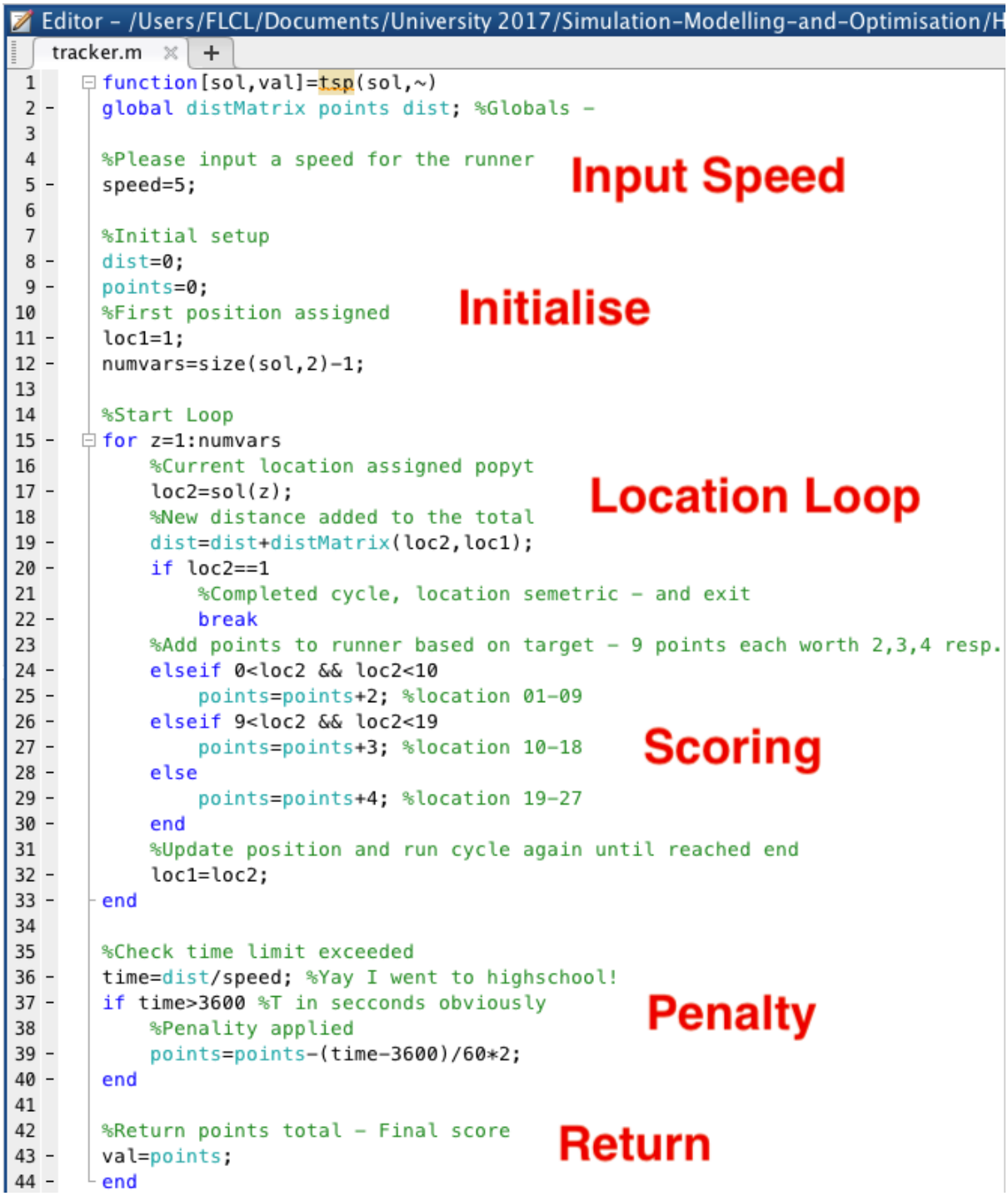


Figure 5: Tracker function, used to check position and calculate points
Note Penalty changed to 30 seconds (stream error before screenshot taken and edited)