COS30018 - Intelligent Systems

Project Report

Topic 2 - Delivery Vehicle Routing System

Group L

Harry Warner 102561680

Lachlan Skinner 101602575

Sarah Howarth 101265321

Table of Contents

Introduction

The system being developed aims to solve the Delivery Vehicle Routing problem through the use of a chosen search optimization technique. The solution implements a Delivery Agent (DA) and Master Routing Agent (MRA) that aims to find the optimal route for each DA so that all parcels will be delivered and the DAs finish their routes at the warehouse. Furthermore, the system also has the extra constraints of vehicle and package weights so that each vehicle never carries more packages than their vehicle capacity during a delivery and aims to have all vehicles leave the depot.

This project report is designed to inform all participants of what is encapsulated within the Delivery Vehicle Routing project. This document also further details the overall system architecture, implemented interaction protocols, search optimization techniques, UI design, scenarios/examples of the system and a critical analysis on the implementation of the system.

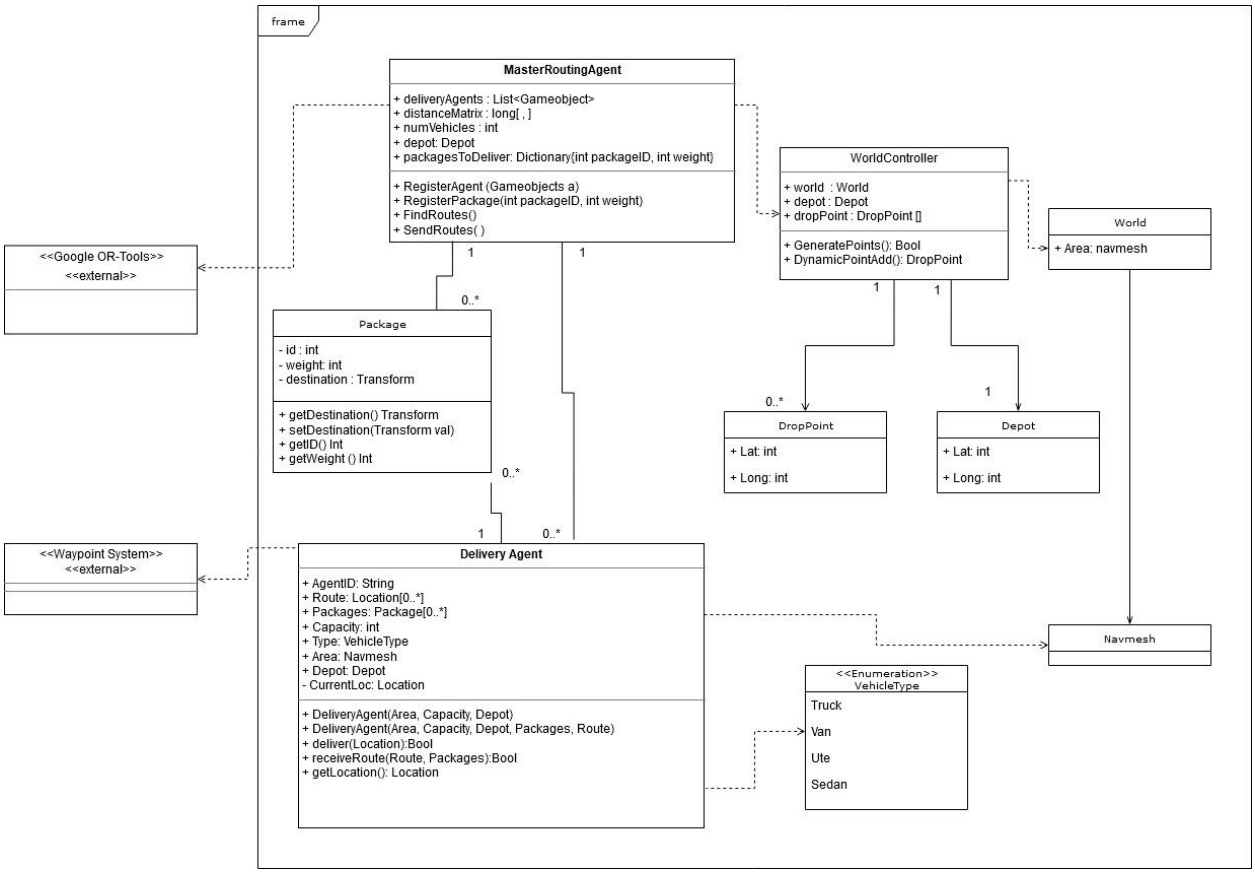
Overall System Architecture

The System follows an Object Orientated design approach with classes for each of the main objects:

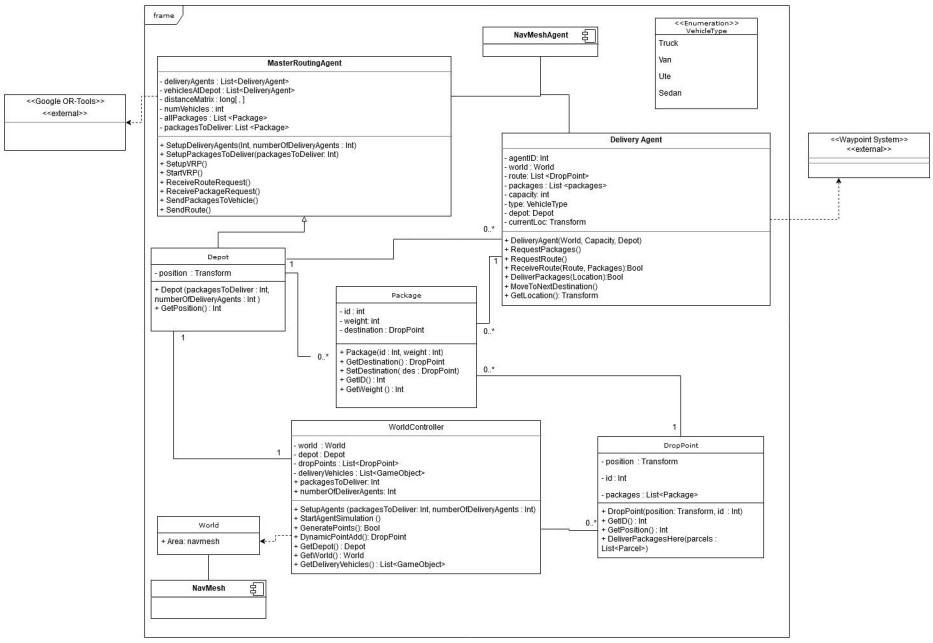
- Master Routing Agent
- Delivery Agent
- World Controller
- Depot
- Drop Point
- Package

UML Class Diagram

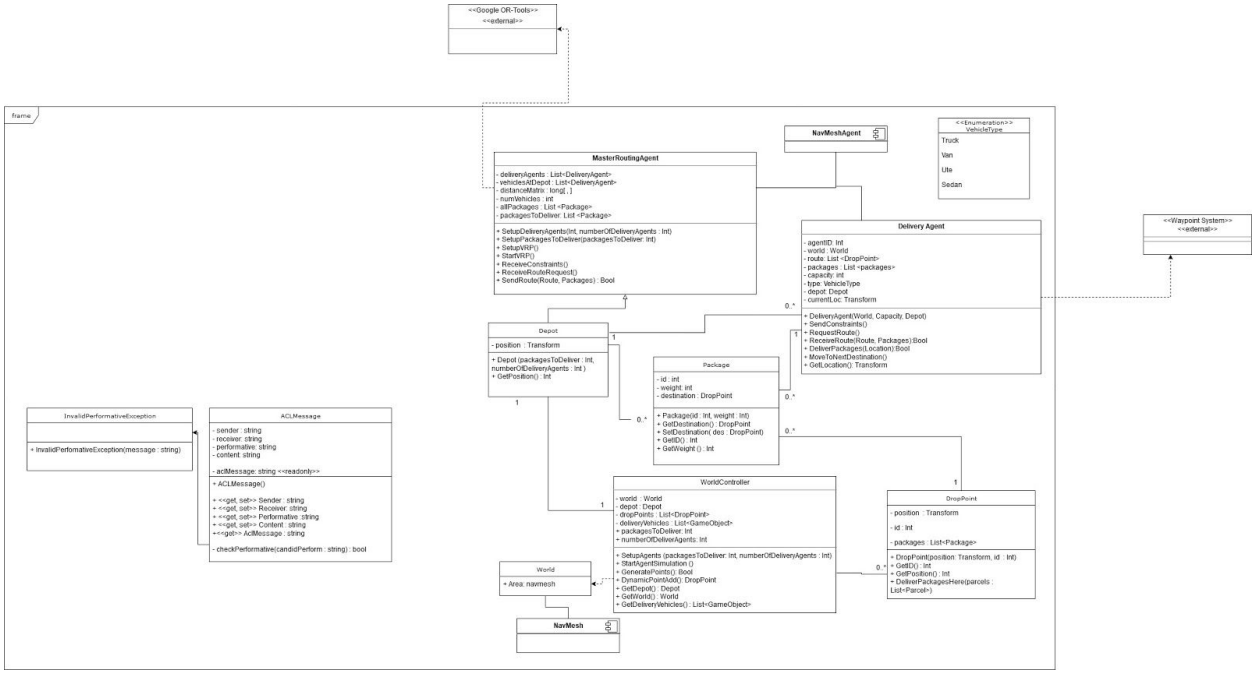
Initial Design:



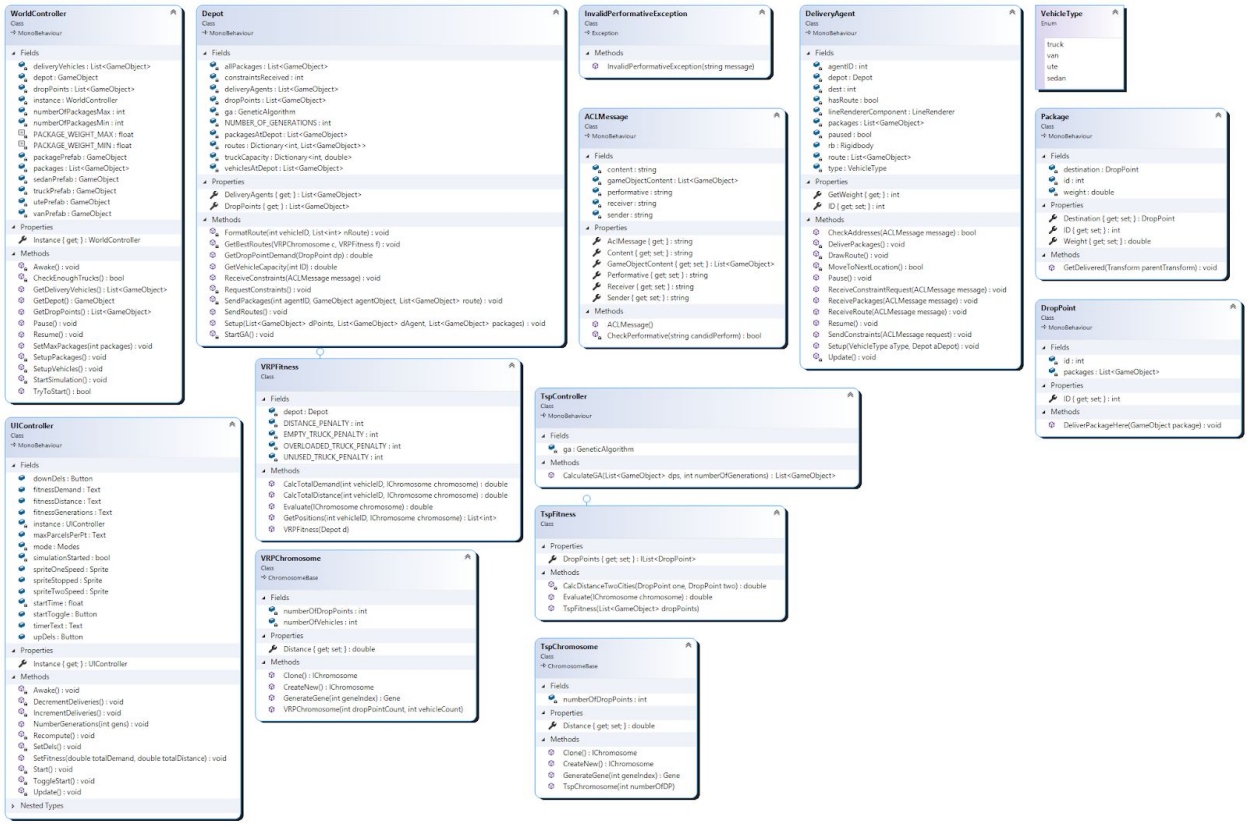
Update 17/09/20:



Updated 24/09/2020:



Final UML Class Diagram:



Object Descriptions

Master Routing Agent & Depot

- The Master Routing Agent began as the Depot's parent class but because we aren't using the MRA for anything else it seemed redundant to have a parent class so we moved all functionality to the Depot itself
- The Depot holds the master routing agent logic required to receive constraints, calculate routes with a GA and send them to delivery agents.

  Google OR-Tools

  Currently used as a substitute search optimisation library

  GeneticSharp

  A Genetic Algorithm C# library that can be used with Unity3D

Delivery Agent

The Delivery Agent class uses a VehicleType enumeration to define the capacity of the agent vehicles, and holds the logic required to send constraints, receive the route with packages, draw their route, deliver packages, and move between drop points.

Currently, the Depot(master routing agent) sends each delivery agent a collection of drop-points - an ordered list of the agent's route destinations. Through this:

- Agents will follow a route when they have one, before returning to the depot, where they will stay until collecting additional routes(if available).
- While the delivery agents are present at the depot, they can engage in route and constraint communications via the ACL message system (see interaction protocols below).

Waypoint System - No Longer Used

Used as a waypoint library to send agents a route

World Controller & World

The World Controller will be a singleton and is essentially the simulation controller; it initializes the simulation and registers/spawns relevant objects. Furthermore, it also communicates with the UIController to set the relevant data from user input; currently max packages for each drop point.

UI Controller

The UI controller is a singleton that operates as the point of control over the simulation. This implements controls on the user interface, allowing the user to change parameters like the maximum-packages-assigned-per-drop-point for extended weight and capacity tweaking, before opting to begin the simulation. The controller also provides access to the features of Unity's inbuilt simulation controls like timeScale; which affects the real-time playback of the simulation and/or its individual elements.

- https://docs.unity3d.com/ScriptReference/Time-timeScale.html

Package

The Package class holds all the data for a package and required property getters. Furthermore the Package class also handles any relevant gameobject functions that need to happen on package delivery; for example moving the game object and any animations if we felt like that was an extension we wanted to go for.

Drop Point

The Drop Point class is essentially a destination that delivery agents can go between and deliver packages to.

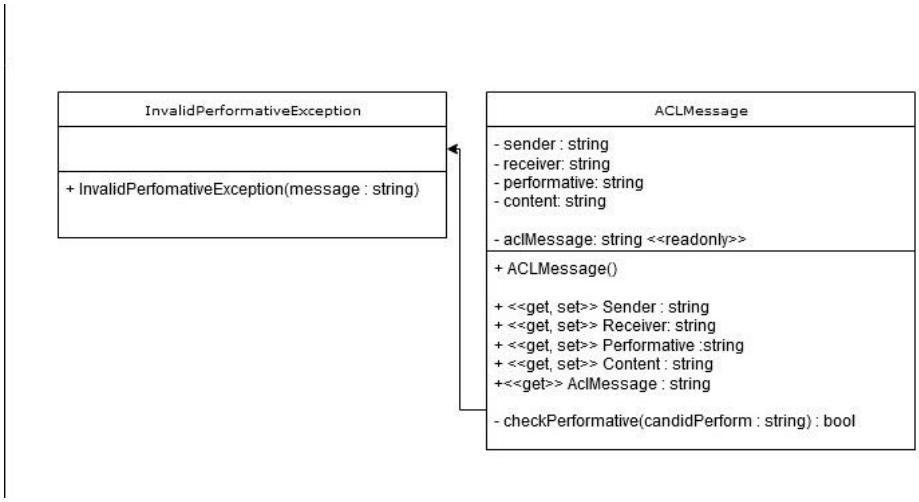
NavMesh & NavMeshAgent - No Longer Used

The NavMesh and NavMeshAgent are Unity3D components that can be used for pathfinding and movement within a game scene

- https://docs.unity3d.com/ScriptReference/AI.NavMesh.html
- https://docs.unity3d.com/ScriptReference/AI.NavMeshAgent.html

Implemented Interaction Protocol

The systems interaction protocol will use a top-down approach; we've designed a set of roles for our agents and a hierarchy within, where the master routing agent will act as a dictator for delivery agents, specifying their role-based-goals in the form of delivery routes.



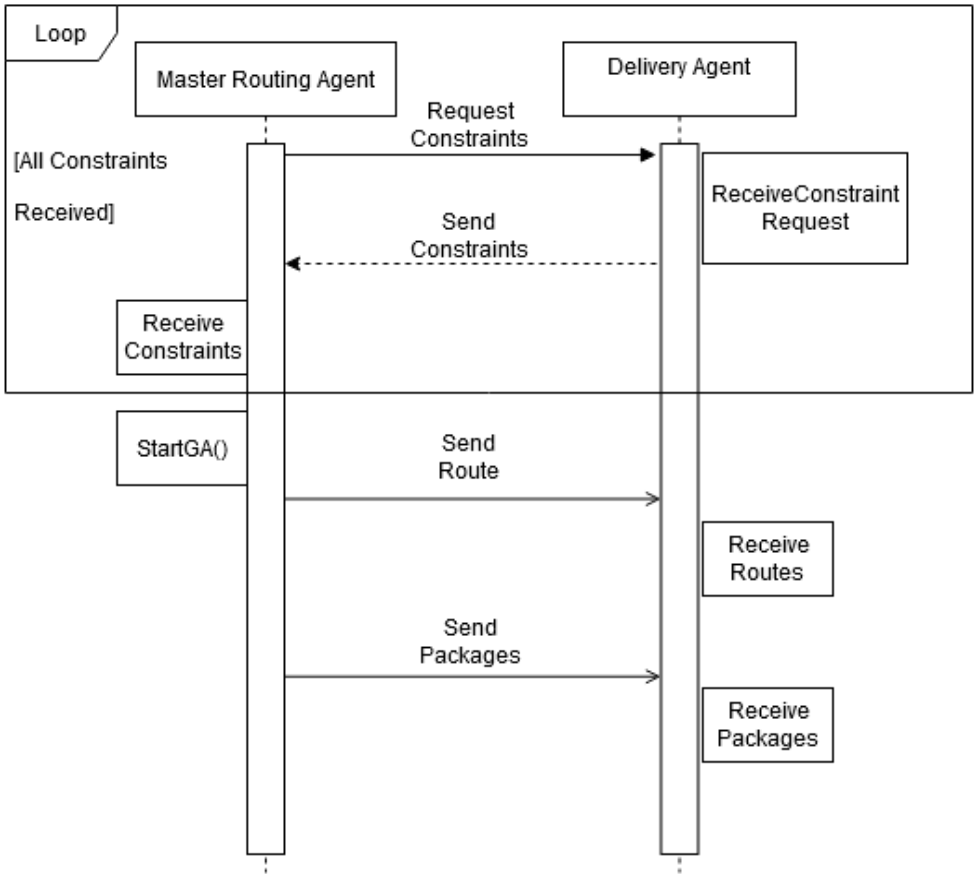
ACL Message Class

As C# doesn't have a specific ACL library, we created an ACL Message class that defines the interaction protocol for the agents and contains properties to define the sender, receiver, performative and content of the message.

- The currently defined performatives: { "request constraints", "send constraints", "send route", "send packages" }
- Also includes a GameObjectContent property for passing the package objects and route objects to the agents.

When the simulation is started and all the relevant data is registered within the depot, the depot constructs and sends a message to each delivery agent that requests they send their constraints back. Once all the constraints are received for the delivery agents the Genetic Algorithm calculates the routes then sends each agent their route and the relevant packages. In the case of the depot, the sender is a simple string "depot" with the receiver being "delivery agent:" and their ID which can be parsed by the receiver to ensure the message is for them. For first implementation, it is a lot easier to just send through the

content as game objects in the route and package messages rather than sending the IDs and having to retrieve the game object with that.



Agent Interaction Sequence Diagram

In order to courier these messages in the simulation system, we've taken advantage of Unity's scripting API sendMessage functions; these allow us to broadcast messages to a specific function in objects that are active in the simulation. These objects, our agents, then decode the message to ensure it's from a recognised sender - relevant to that function - and that it's a message addressed to them before accessing the message's performative and content components.

- https://docs.unity3d.com/ScriptReference/GameObject.SendMessage.html

Implemented Search Optimization Technique

As the system uses C#, the majority of resources provided by the unit proved unhelpful for implementing the search optimization technique. So, we did initial research into C# libraries for optimization algorithms in tandem with choosing the best option for us from constraint programming, ant colony optimization, particle swarm optimization and genetic algorithms. The first implementation looked at using Google OR-Tools as they provide helpful resources for implementation and support C#. But this proved redundant as we needed to find our own search optimization technique to implement so we started looking at more libraries that provided support for C#.

The search optimization technique we eventually chose to use is a Genetic Algorithm (GA) as it is seen as an efficient technique to solve highly complex problems and allows for scalability if the vehicle size increases (Mohammed, Ahmad & Mostafa 2012). Furthermore, C# Unity3D is fairly scarce with AI libraries but GeneticSharp provides a GA library that can be used within Unity3D.

A TSP algorithm was first implemented to get an understanding of GA and how it would work in the Capacitated Vehicle Routing Problem. This then allowed us to implement a GA with the following properties by default.

Chromosome Representation

There are several ways chromosomes can be represented in a capacitated vehicle routing problem for example encoding each chromosome as a list of lists that represents each gene as a list of routes for a vehicle. This implementation proved too difficult to implement with Genetic Sharp as embedded lists within a gene are not supported.

With the current system, each chromosome represents a list of genes the size of drop points that have packages. The number encoded into the genes represent the vehicle ID that will service that location.

Example for 5 vehicles and 10 drop points:

Chromosome	0	2	1	4	3	3	1	2	3	4
Chromosome	0	0	1	3	1	2	2	4	4	3

Population Initialisation

The initial population for the genetic algorithm is generated randomly; the current population size varies from 150-200.

Fitness Function

The fitness measure for evaluating each chromosome is the total travel distance and the total demand. Furthermore, the fitness function also takes into account the following constraints and calculates a total demand cost:

- If a truck is overloaded, the amount the truck is overloaded by is multiplied by 200
- If a truck has unused weight, the amount that is unused is multiplied by 20
- If a truck is empty, the vehicle's capacity is multiplied by 80

The fitness is then calculated by multiplying the distance penalty by the total distance and then taking the total demand cost away from that, so to increase fitness in chromosomes that haven't been penalized:

- Fitness += (Distance_Penalty * totalDistance) - totaldemandCost

Crossover Operators

The GA uses Two Point Crossover where the operator calls for two points to be selected on the parents. Furthermore, everything between the two points is swapped between the parents, rendering two children.

Mutation Operators

The GA uses Reverse Sequence Mutation where we take a sequence S limited by two positions i and j randomly chosen, such that i<j. The gene order in this sequence will be reversed by the same way as what has been covered in the previous operation.

Selection Operators

Two selection operators have been chosen for testing the optimum solution:

1. Roulette Wheel Selection which is a kind of Fitness Proportionate Selection. The first step is to calculate the cumulative fitness of the whole population through the sum of the fitness of all individuals. After that, the probability of selection is calculated for each individual. Then, an array is built containing cumulative probabilities of the individuals. So, n random numbers are generated in the range 0 to fitness sum. and for each random number an array element which can have higher value is searched for. Therefore, individuals are selected according to their probabilities of selection.
2. Elite Selection which chooses the chromosomes with the best fitness.

Termination Condition

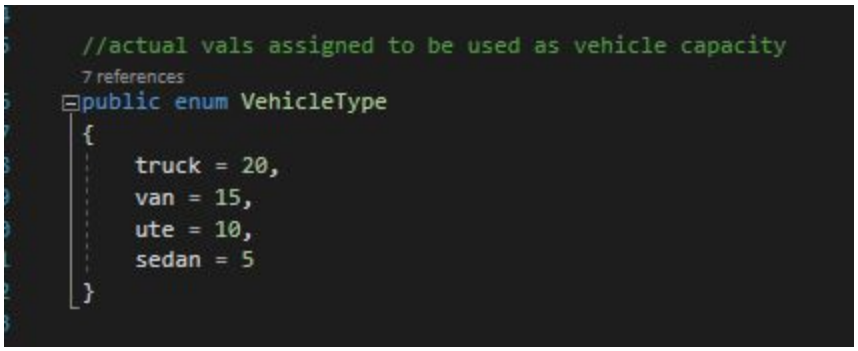
The GA terminates when it hits a number of generations.

Alternative 1: Specifying Vehicle Capacity with Weight

Through the GA fitness evaluation the system ensures:

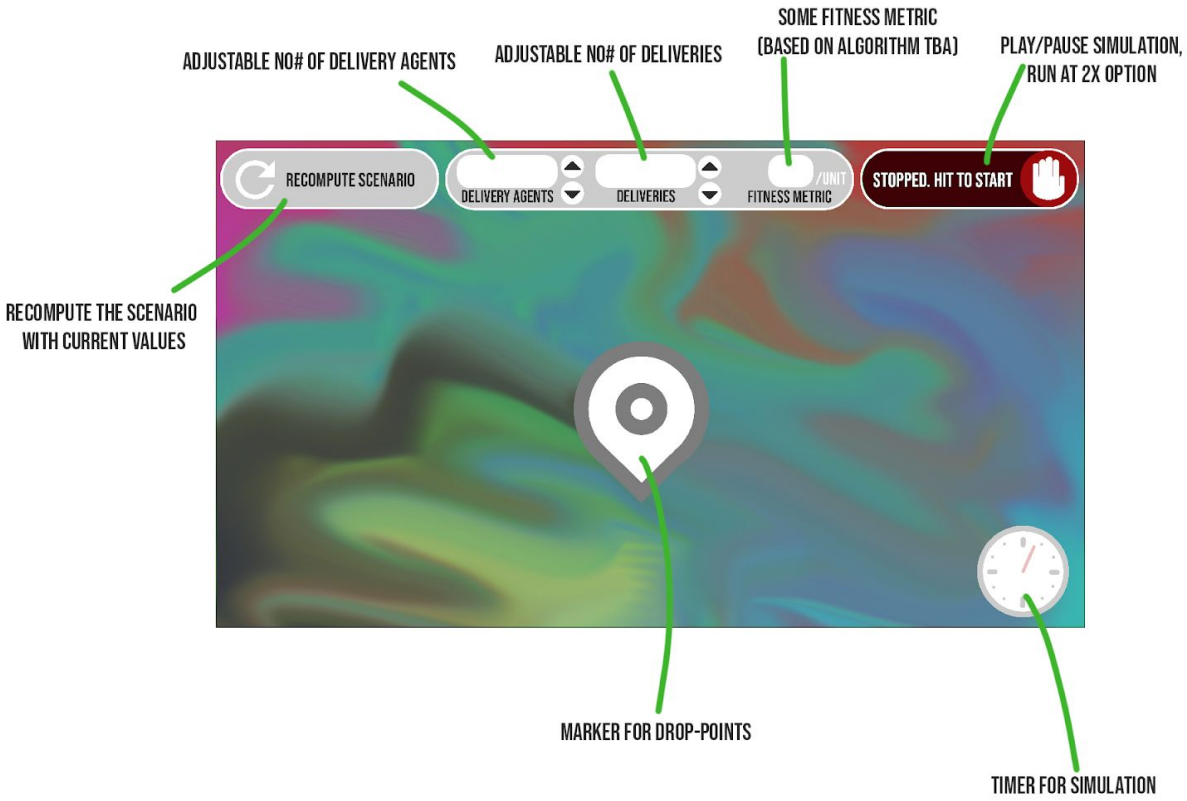
- The total weight of all items to be carried by the vehicle does not exceed the vehicle's capacity
- The vehicle is not empty

Each delivery agent has a truck type which is stored as an integer and then this is parsed as a float due to the nature of enums only being able to hold integers. Furthermore, the packages are assigned a random weight between two min and max floating point values assigned in code.

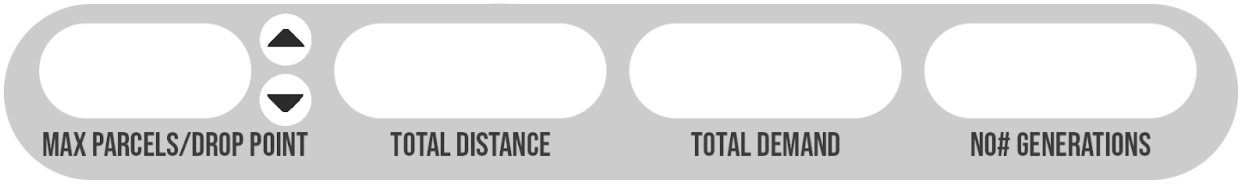


Truck Type Enum

User Interface Interactions - UI

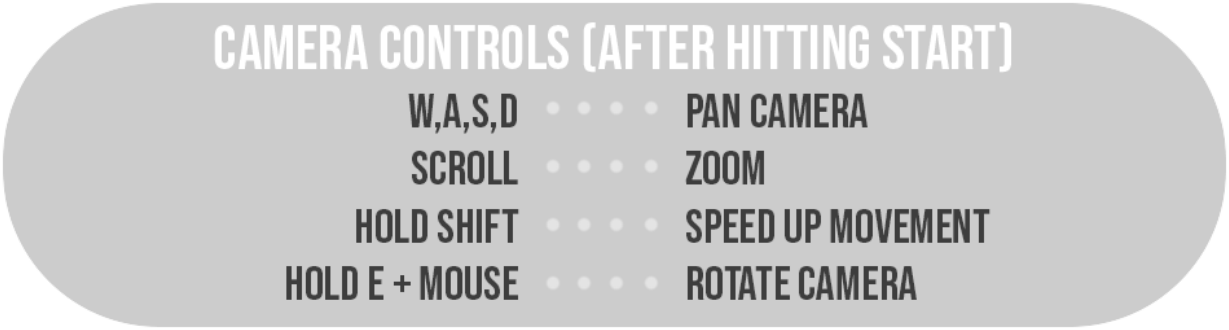


Version 1.



Center Bar, updated 25/10/2020 (to change adjustable values and fitness metric areas).

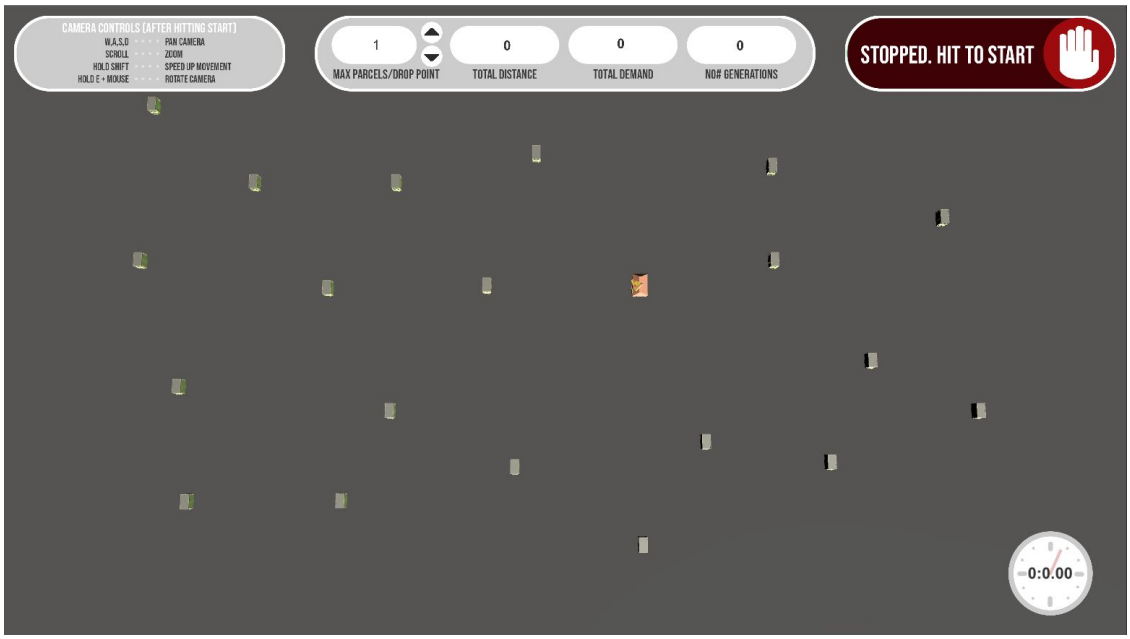
The UI serves to allow the user to adjust simulation parameters, before observing genetic algorithm fitness characteristics under the influences of a scalable time-speed toggle.



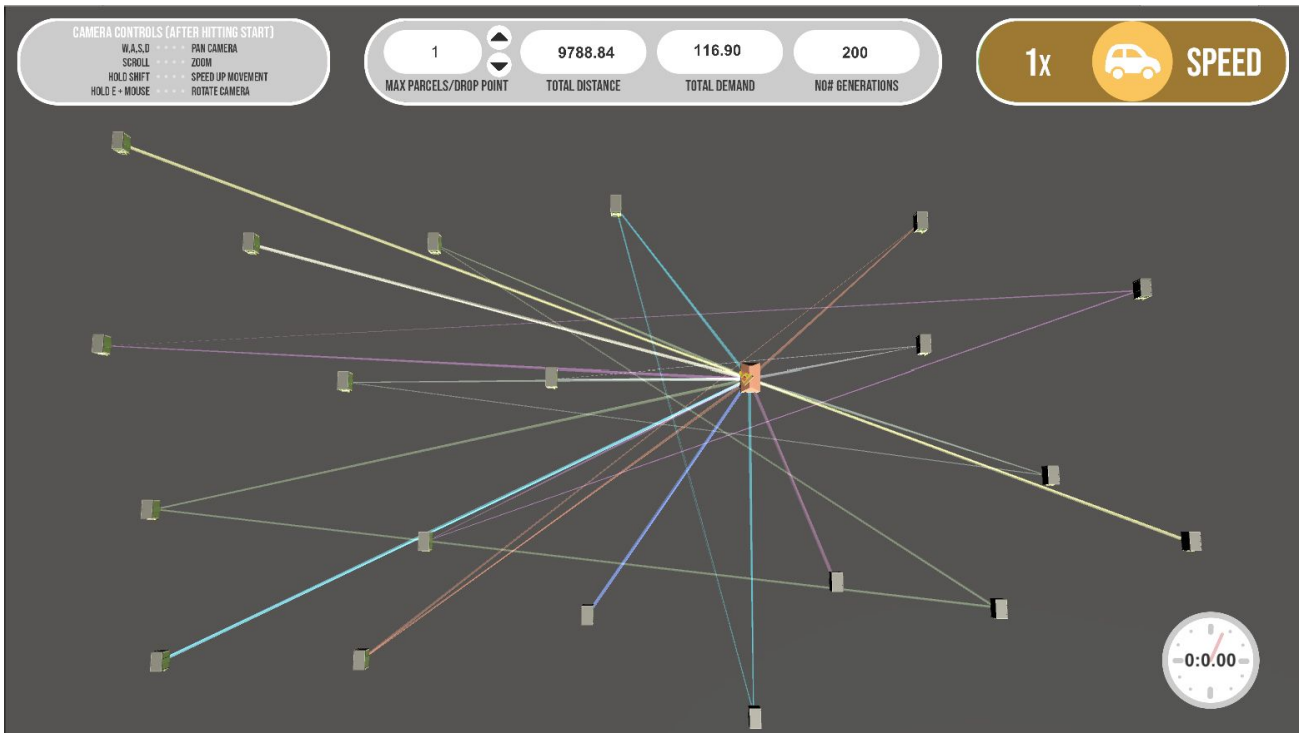
UI update, updated 30/10/2020, camera controls implemented with a key.

This will take the space of the recompute button, as recomputing/reinitialising will be automatic after the simulation concludes.

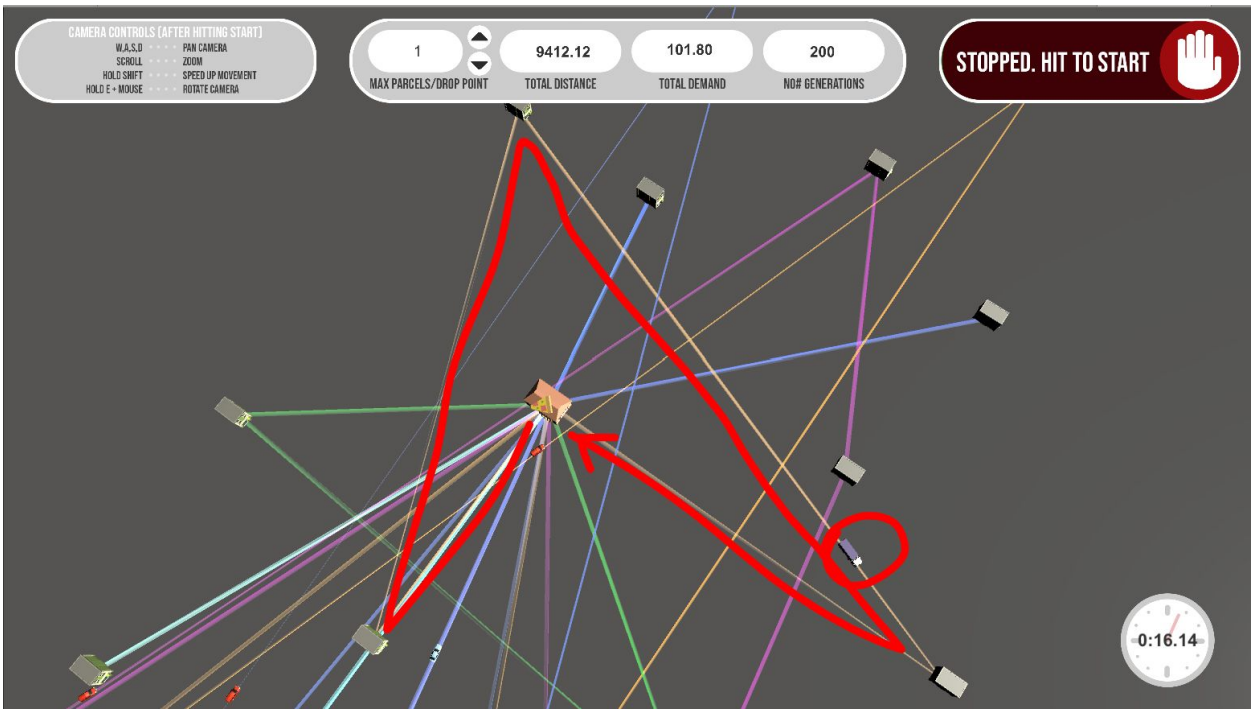
Scenarios/Examples of System Running



Initial state: The system prepares the depot and dropPoints, here the user can adjust the maximum parcels assigned per drop points before the algorithm runs, in order to test vehicle capacity/weight functions.



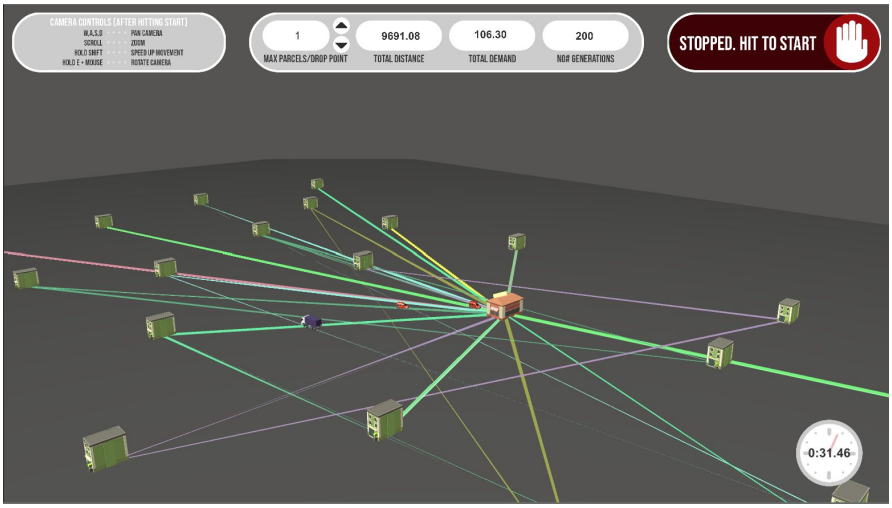
User-Selects Start: Upon starting, the algorithm takes a moment to compute optimal solutions over (currently)200 generations, taking into account the distance-per-vehicle, and the capacity for weighted packages each type of vehicle has. While computing, the depot sends messages to identify the capacity of each delivery agent and, when the generation threshold is hit, sends messages to each delivery agent containing a formatted route and collection of packages.



Capacity Routing Example: Denoted alongside red annotations, this brown route is quite long due to a big spread of dropPoints. What's important to notice is that the circled vehicle is a truck - the highest capacity vehicle(camera can be further zoomed), and this unique route shape is a result of those 3 dropPoints having a large number of packages and overall weight demands. This truck, the most equipped to handle big deliveries, takes on the heavier packages in order to free up what would be a solo trip for some of the smaller vehicles.

Return:

Upon delivering all packages, each delivery agent returns to the depot, before the simulation starts again.



Critical Analysis of Implementation

The solution works to find the most optimized route for each truck while still remaining within constraints but it fails at making routes that you would deem 'the best', producing routes that sometimes go from each end of the map rather than staying in a small cluster of drop points. The penalties work to aim at not overloading trucks but the system would benefit from a map of connected nodes in order to produce shorter and 'prettier' routes. Furthermore, the GA finds an optimum solution but it would have been good to implement a better suited crossover operator that dealt specifically with routing based problems rather than just using the basic provided operators. Moreover, the population initialisation could have been handled better in order to find a more optimum solution faster as randomising the initial population isn't always seen as the best option.

- Would have been good to implement the TSP or Google OR tools as a benchmark to see if the routes being produced were optimum

A good way to benchmark the system to see how well it achieved its routes etc. would have been to implement the TSP genetic algorithm for the routes once the packages have been assigned which would have allowed us to find the most optimum route with our system that was already doing well with truck capacity constraints. Moreover, implementing Google OR-Tools would have helped with finding out whether or not our GA was performing better than the A* pathfinding and constraint programming that OR-Tools uses.

The system is minimal so a good addition would be the ability for the user to dynamically add drop points and packages on the fly and for the system to be able to recompute and still provide optimum solutions. Furthermore, as the system does delay when the genetic algorithm is computing, multithreading would allow for us to implement this and not have a delay on runtime. But, this has proved to be out of scope due to the further research needed to actually understand the genetic algorithm implementation and extra programming needed to handle deletion of game objects and multithreading.

Summary/Conclusion

This project has been extremely rewarding for our group in the context of intelligent systems. Not only have we been able to learn through the planning and development of the delivery routing system, but have also been able to implement a genetic algorithm to provide solutions to this environment.

Through clear Intelligent agent roles, and a structured messaging system, our system's agents are able to successfully, and somewhat efficiently , route and execute delivery paths in an environment that demands adaptiveness in each cycle of execution.

While there are improvements that could be made to the project with more time and resources, the outcome we've reached through valuable research and practice fulfills the project brief.

References

Awad, H, Elshaer, R, AbdElmo'ez, A & Nawara, G 2018, 'An Effective Genetic Algorithm for Capacitated Vehicle Routing Problem', *Proceedings of the International Conference on Industrial Engineering and Operations Management,* Indonesia, pp. 374-384.

Mohammed, M, Ahmad, M & Mostafa, S 2012, 'Using Genetic Algorithm in Implementing Capacitated Vehicle Routing Problem', *2012 International Conference on Computer & Information Science (ICCIS)*, pp. 257-262.

NEO 2013, *Genetic Algorithm*, NEO, viewed 1st October 2020, <https://neo.lcc.uma.es/vrp/solution-methods/metaheuristics/genetic-algorithm/>.

Shabir, S & Singla, R 2016, 'A Comparative Study of Genetic Algorithm and the Particle Swarm Optimization', *International Journal of Electrical Engineering*, vol. 9, no. 2, pp. 215-223.

Shahab, M, Utomo, D & Irawan, M 2016, 'Decomposing and solving capacitated vehicle routing problem (CVRP) using two-step genetic algorithm (GA)', *Journal of Theoretical and Applied Information Technology*, vol. 87, no. 3, pp. 461-468.

Unity Technologies 2020, *Unity Documentation - Scripting API,* Unity, viewed 10 September 2020, <https://docs.unity3d.com/ScriptReference/index.html>.

Tools/Libraries/Assets

Genetic Sharp <https://github.com/giacomelli/GeneticSharp>

Google OR-Tools <https://developers.google.com/optimization>

Unity3d 2019.3.2f1 <https://unity.com/>

3D Car Models
<https://assetstore.unity.com/packages/3d/vehicles/land/low-poly-cars-101798>

3D Building Models

<https://assetstore.unity.com/packages/3d/simple-town-lite-cartoon-assets-43480>

End of Semester Final Deliverables Check List

The following documents must be included in the final submission by the due date.

Deliverables Check List	Submitted Yes/No/NA
Main Document	
1. Cover Sheet	Yes
2. Check List	Yes
3. Project Report	Yes
Appendices	
4. Source code and executable - Linux and Windows Build	Yes
5. Who did what	Yes
Others	
6. Presentation slides and A0 Poster (submitted a day before the presentation)	*Don't Need This Semester - 10 min presentation due

COS30018 Intelligent Systems

Assignment Deliverable Guide

Cover Sheet Information

Project Title: Topic 2 - Delivery Vehicle Routing System		
Team Name: Group L		
	ID Number	*Name*
1	102561680	Harry Warner
2	101602575	Lachlan Skinner
3	101265321	Sarah Howarth
4		
5		
6		

COS30018 Intelligent Systems

Assignment Deliverable Guide

Who Did What.

As you are to be individually assessed it is necessary to ensure your marker understands your individual contribution.

This document is to demonstrate who was responsible for each piece or contribution to each piece of work in your project.

The following is a template to present and **must be signed by all team members**.

Project Title	
Team Member	Activity (these are suggestions of the ways you may have contributed. There may be others and some may not be applicable). You may also add comments to further explain your contribution or partial contribution to an activity.
Name: Harry Warner Student No. 102561680	Research Details - Assisted in researching the different algorithms to decide which algorithm to use - Researched the Travelling Salesman Problem to understand the challenge Software Design/Coding Details - ACL Message class - Invalid Performative Exception class - DropPoint class - Package class - Created documentation - Co-developed the UML Diagram - Debugging / Fixing errors in code Project Management Details - Delegated and completed weekly tasks using Trello, alongside both other team members

	Presentation Details - Prepared resources for use in the presentation. - Presented

COS30018 Intelligent Systems

Assignment Deliverable Guide

Name :

Lachlan Skinner

Student No.

101602575

Report

- Contributed to population, editing, and visual assets

Sections

- Overall system architecture
  - UML class diagrams
  - Object Descriptions
- Implemented interaction protocol
- User Interface Interactions
- Scenarios/examples of System running
- Critical analysis of Implementation
- Summary/Conclusion
- References

Research

Details

- Research into Unity Scripting API, for the support of messaging systems and the development of the simulation environment.
- Assisted in research into possible search optimisation techniques
- Research into existing routing software solutions to observe fitness changes with user input parameters.
- Research into existing C# implementations of delivery routing problems.
- Research into GA implementations, and characteristics used for fitness in Delivery routing problems.
- Assisted in research into Google-OR tools, particularly the context of TSP.

Algorithms

Details

- Assisted planning and discussion for Algorithm, while actual implementation led by Sarah.

Software Design/Coding

Details

- Planning and design discussion for implementations of the project with the team.
- Created the UI for the application
- Co-developed UML diagrams
- Discussion and planning for algorithm communications
- Implementing Interaction protocol methods for the Delivery agent, and updates the Master routing agent interactions.
- Developed the DeliveryAgent script
- Developed the UIController
- VehicleType enumeration
- extracting GA fitness characteristics to display to user
- QA and debugging throughout the whole project
- CameraControls script

Project Management

Details

	- Organised group method of communication (Discord server) - Prepared collaborative cloud storage through google drive - Regular discussion and planning for the project and deliverables with the team. - Organisation of presentation script outline and delegation of parts between team members - Delegated and completed weekly tasks using Trello, alongside both other team members Testing - Regular testing throughout the progression for the project. Presentation - Prepared outline for group - resources for use in the presentation. - Presented

COS30018 Intelligent Systems

Assignment Deliverable Guide

Name : Sarah Howarth Student No. 101265321	Report - Setup the Report Document and initial template Sections - Introduction - Overall System Architecture - Implemented Interaction Protocol - Implemented Search Optimization Technique - Alternative 1: Specifying Vehicle Capacity with Weight - Critical Analysis of Implementation - References - Tools/Libraries/Assets Research Details - Research into the implementation of Google OR-Tools and how they address the CVRP - Assisted in research into possible search optimization techniques - Research into C# libraries that support the possible search optimization techniques - Research on Genetic Algorithms (GA) and how they are implemented in VRP and CVRP - The best crossover, selection and mutation operators that could be used - How chromosomes are represented in a CVRP - Further research into fitness functions in GA's specifically CVRP - Research on the TSP and GA's to further understand how the optimization problem works - Provided references for the report that pertain to genetic algorithms Algorithms Details - Implemented GA Software Design/Coding Details

	- Integrated Genetic Sharp in the project - Integrated Google OR-Tools in the project - Package Class: GetDelivered Function - DropPoint Class: DeliverPackageHere Function - DeliveryAgent Class: Drawing of route - ACLMessage Class: Add gameObjectContent property and variable - TspFitness Class - TspController Class - TspChromosome Class - WorldController Class - VRPChromosome Class - Depot Class - VRP Fitness Class - Setting up the Unity scene - Setting up the prefabs for packages, the different trucks, depot and drop points - Planned, created, and implemented parts of the UML Diagram - Created final UML diagram from Visual Studio - Created and updated the interaction protocol sequence diagram - Added scene changing and loading of new scene when simulation done - Created Builds for submission Project Management Details - Created Trello board and invited team - Created GitHub and invited team - Delegated and complete weekly tasks using Trello, alongside both other team members - Updated tutor via email with the teams progress Testing Details - Debugging/fixing errors in code - Building of the project and testing - Testing of the GA to ensure output is correct - Testing of the GA to ensure the penalties are correct - Testing of the project to find better vehicle number and package weight values

COS30018 Intelligent Systems

Assignment Deliverable Guide

	Presentation Details - Prepared resources for use in the presentation - Created some of the script for the presentation - Created slides for the presentation - Edited video presentation with Premiere Pro

I declare this is an accurate description of team contributions of the team members

Team Member Name	Signature	Date
Harry Warner		28/10/20
Lachlan Skinner		30/10/20
Sarah Howarth		1/11/20