Software Requirements Specification

for

<Multi-agent reinforcement learning of traffic routing in networks of autonomous vehicles>

Version 1.0 approved

Prepared by <Alon Reicher, Liad Gam>

<Bar Ilan>

<21/11/22>

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

To develop a multi-agent learning algorithm for traffic management of autonomous vehicles in an urban network.

## Intended Audience and Reading Suggestions

This document is intended for customers and for developers who will use the system.

## Product Scope

The product will include a simulation of urban traffic while integrating models of the effect of load on traffic speed.

Utilizing the simulator, the user could run a MARL algorithm for cooperative learning of vehicle routing while using information from other vehicles.

## References

[1] A. Leshem, Y. Sung et al., “Deep multi-agent reinforcement learning of Markov games for distributed traffic management in smart cities”, 2022.

[2] G.F. Newell, "A simplified car-following theory: a lower order model", Transportation Research Part B: Methodological, Volume 36, Issue 3, 2002.

**References we could use in the project**

* Polaris Traffic Modeling, youtube (nice simulation)
* Newell's car-following model
* Gipps' model (mathematical equation about car acceleration)

# Overall Description

## Product Functions

* Running a traffic modeling simulation based of a user-controlled parameters.
* Visual presentation of the traffic.
* Applying a multi agent RL to achieve a better traffic flow in the simulation.

## User Classes and Characteristics

The users that will be using this product are:

* The developers that could expand the product for broader use cases.
* Traffic researcher to use different models and to understand the behavior of the traffic.
* E-commercial companies to manage logistics within their facilities.

## Operating Environment

Python in a PC + a server.

## Design and Implementation Constraints

TBD

## User Documentation

TBD

## Assumptions and Dependencies

TBD

# External Interface Requirements

## User Interfaces

* Visual simulation of the traffic flow.
* Changing simulation parameters in real time.
* View simulation statistics.
* Loading a learning model to the agents in the simulation.
* Adjusting single agent learning parameters.

## Software Interfaces

* Database-
* Python IDE-
* Python libraries-

## Communications Interfaces

TBD

# System Features

## Traffic flow simulation

4.1.1 Description and Priority

A simulator that will simulate traffic modeling based of: number of cars, road's characteristics, weather and etc.

This feature is of high priority.

Requirements:

1. Road's parameters
   1. Type (e.g., highway)
   2. Length
   3. Number of lanes
   4. Speed limit
   5. Maximum flow rate (VPHPL- Vehicles per lane per hour)
   6. Current flow rate
   7. Density (VPKPL- Vehicles per kilometer per lane)
   8. Road's condition (weather, construction, accidents, etc.)
   9. Public transportation lanes
2. Agent's parameters
   1. Drive time
   2. Speed
   3. Car's speed limit
   4. Acceleration/Deceleration
   5. Starting point
   6. Destination
   7. Route
   8. Size of car
   9. Agent's scalability
   10. Current road
3. Graph's requirements
   1. Scalability
   2. Transforming real life maps into graph

## Visual representation of simulation

4.2.1 Description and Priority

A 2D representation of the simulation. This feature will give an intuitive understanding of the simulation to the user.

This feature is of medium priority.

1. Statistical requirements
   1. Agent's speed over time
   2. Agent's Distance over time
   3. Average drive time
   4. Road's speed over time
   5. Road's Distance over time
   6. Number of cars in a road over time
   7. Road's density over time
   8. Matching time scale to simulation time
2. Map visualization
   1. Heat map representing road's density
   2. Road's condition
   3. Agent's route (alternative route + original route)
   4. Ramp meters
   5. Zoom in/out
3. Interface requirements
   1. Option to toggle/change road's and agent's parameters
   2. Setting the simulation's parameters
   3. Option to slow down/fast forward the simulation

## Multi agent reinforcement learning

4.3.1 Description and Priority

This feature will apply a reinforcement learning algorithm to the vehicles in the simulation. It will allow them to learn, using the algorithm, how to navigate faster using multi agent learning.

This feature is of high priority. l

1. Penalty for agents using toll roads.
2. Creating sub goals for agents.
3. Collaborative communication of the agents.
4. Learning optimal path finding strategies under communication limitations.
5. Scalable network optimization via MARL with low complexity.
6. Exploiting the periodicity of the transportation.

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## ~~Safety Requirements~~

TBD

## ~~Security Requirements~~

TBD

## ~~Software Quality Attributes~~

TBD

## ~~Business Rules~~

TBD

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

**Annex**

**Ground Assumptions**

**Traffic axioms**

* Every car that enters the road will also leave it.
* Every car has an origin and a destination.The assumption is that there is one or more paths from origin to destination exists.
* Every start and end points are vertices. We assume that a vertex is connected to 2 edges at least, and an edge is a road (2 edges vertex- dead end: edge in and edge out).
* Car won't stop moving towards its end goal.
* Every edge has a certain number of cars in the free flow area. There will be a queue for the free flow area to determine cars' speed. The speed is the same for all cars.

**Agents' axioms**

* decision of next edge must be before previous edge turning queue at latest.
* Every "smart" agent in the simulator has the same thought process a smart agent could dynamically change his path according to information from other smart agents. Other "dumb" agents will determine an initial path and won't change it.

**make a power point animation of a graph of 2 possible car flows. Useful to understand, animation is very helpful.**

**Map generating method**

* The user will write as an input the required number of intersections (nodes).
* The simulator will generate a graph including the said number of nodes and random edges.
* The user will be able to save a map and load it.

\*Maybe not only random models

* The roads graph will be a directed graph, and every edge will have an "opposite" which has the same basic stats (length, max speed etc…) but is in the opposite direction.
* Each node will have a maximum of 4 next edges and minimum 1 next edge (representing real life junction).
* Edge is made of queues of each direction, and a single-lane movement part. The queues will be at the end of the edge, representing the stopping line before the junction. The queue is infinite.

There will be a different queue for every next immediate edge (one per edge).

**Modeling the free-flow area**

**LWR Model**

LWR model is a mathematical model that describes the flow of traffic on a road.

Where is:

- flow rate (number of vehicles/ units of time)

- density (number of vehicles/ distance)

- speed (distance/ units of time)

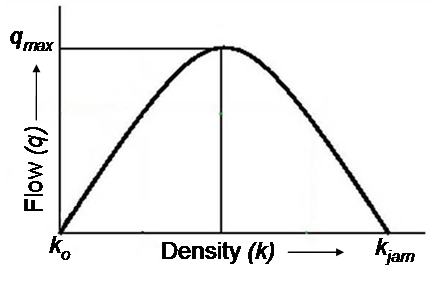
It is based on the assumptions that the speed of a vehicle on a road is determined by the speed of the vehicle in front of it, and that the spacing between vehicles is determined by their speeds and accelerations.

The model is based on the following equation, which describes the speed of a vehicle as a function of the speed of the vehicle in front of it:

The equation must fill 3 conditions:



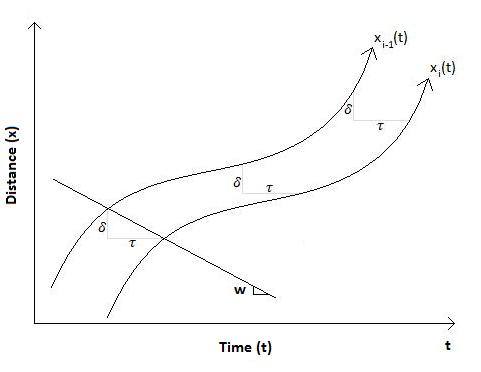
We can place this in the initial equation, and we get:



If we derive this equation by to find the maximum flow, we find that the maximum flow rate happens when .

**Newell's Model**

Newell's model describes how a vehicle will follow another vehicle in traffic while maintaining a minimum space and a time gap to the vehicle ahead.



- the spacing headway (in unit of m).

- the speed (in unit of m/s) of the vehicle n.

- the driver's reaction time plus the vehicle response time.

- the minimum safety spacing (in unit of m) the follower wants to maintain in stationary traffic when the speed is equal to zero.