**CHAPTER ONE**

**INTRODUCTION**

* 1. **Background to the Study**

Vehicle routing problems (VRP) belong to a class of path planning problems and has been studied for more than five decades. (Ezzatollah et.al, 2022) affirmed that VRP is one of the critical challenges that exist in logistics. Dantzig and Ramser were the pioneers of VRP in 1959. Path planning is a classical optimization problem with several of its derived solutions adapted in telecommunication, transportation, logistics and gaming. Usually, an investigator or a user proffering a path planning solution tries to select the most efficient and viable sequence of subpoints that guarantees a fair trail from one to another. Sometimes the navigation could be carried out by a learning agent or driverless car that tries to learn how best to navigate along the road connections under some static or dynamic conditions. Under static conditions, an agent maybe configured to always follow the usual shortest route to a destination but that may not be the shortest route when the conditions are dynamic and sometimes skittish. Traffic management is complex and requires that a learning agent should be somewhat autonomous enroute to its destination.

Traffic congestion has become a daily nightmare due to the growing number of vehicular movement and a lean road network capacity. (Marlin et.al, 2018) reported that daily delay hours grew from 2.5 billion in 1995 to about 4.2 billion in 2005 in the United States of America. Dutch organization for logistics confirmed that about 10% of lost hours for the truck Drivers are owing to traffic congestion. Traffic congestion is very costly for services that depend on the road, most especially logistics and distribution firms. This undesirable condition has increased the overhead cost for business operations such as hiring more truck Drivers to meet up with the delivery targets or being penalized for violating driving hour regulation. Therefore, any conscientious effort to drastically reduce delay time which comes with some cost savings for the businesses is desirable and necessary. Traffic congestion maybe caused by large number of commuters at peak hours, vehicular breakdown, accidents, and sometimes bad weather.

(Kok et.al, 2012) proposed a Time Dependent Shortest Path Problem (TDSPP) and reported that delay hours are predictable, and they account for the largest portion of all traffic congestion. (Orda et.al, 1990) solved TDSPP using a modified Dijkstra search. (Alizadeh et.al, 2020; Ezzatollah et.al, 2022) implemented different strategies for Green Time Dependent Vehicle Routing Problem (GTDVRP). (Bentner et.al, 2001; Scheider,2002) proposed a model for Time Dependent Travel Salesman Problem (TDTSP). Solutions of TDSPP, TDTSP and GTDVRP are solutions of VRP. (Van-Woensel et.al,2008) implemented a time dependent VRP under dynamic conditions.

It is noteworthy that models and algorithmic principles are not enough to ultimately solve the problem of delay due to traffic congestion, government must enact and enforce policies, open more road networks, educate the citizenry on safe road ethics, and implement systems that can suggest alternative routes to Drivers or learning agents.

The project proposes a model that predicts the best possible route from one point to another based on the proposed road network as the case study. This project belongs to the class of time dependent dynamic vehicle routing.

* 1. **Statement of the Problem**

Time is a constraint on most human activities and time delay is never desirable. Usually, commuters are limited in their ability to determine the shortest path to a location based on the current traffic conditions, sometimes the guess of a shortest route could be right or wrong at other times. The problem is to develop a strategy or model that suggests the most efficient route from one point to another point within the proposed road network when there is traffic congestion.

* 1. **Aim and Objective**

The objective is given a network arrangement, the identified model or algorithm should be able to suggest the best route to a destination.

The aim of this project is to design and implement a dynamic time dependent vehicle routing model to determining the shortest path in the proposed road network.

* 1. **Research Methodology**

The road network is modeled as a Markov decision Process (MDP) and Q-Learning algorithm with adaptive reward scheme is proposed to optimize the possible solutions. MDP has been applied in many computational problems especially combinatorial problems. MDP is a tuple parameter-based model; it involves the learning agent, state spaces, set of actions, reward scheme, policy (rule for selection) and production (action-state mappings).

* Build the model of the road network
* Generate random flow rate (speed) and distances along each road connection
* Use Q-Learning algorithm to optimize state transition decision
* Implement an adaptive reward scheme
* Select route with the highest objective value
* Implement solution on Azure Machine Learning Studio
  1. **Research Question**
* Can an algorithm suggest a more favourable route when there is traffic congestion?
  1. **Scope of study**

The scope of this project would revolve around the proposed road network and the implementation would be done on Azure Machine Learning Studio. It will not involve any hardware.

* 1. **Significance of the Study**

This research will enhance the navigation judgments of commuters when there is heavy traffic load on usually optimal paths to take other paths that are not as optimal but having lower traffic costs to the same destination. The system can be fully commercialized for better traffic management or decision.

* 1. **Project Outline**

Chapter two will be dedicated to the literature review on up-till-date account on vehicle routing problems and solutions. Detail analysis on the road network formation, problem modeling in Markov decision problem (MDP) and full methodology will be reported in chapter three. Chapter four will cover the actual implementation, tools used, results and further analysis on the results. Chapter five will cover the recommendation, challenges, and future work in line with the completed project.

* 1. **References**

Alizadeh-Foroutan, R., Rezaeian, J., and Mahdavi, I. (2020). “Green Vehicle Routing and Scheduling Problem with Heterogeneous Fleet Including Reverse Logistics in the Form of Collecting Returned Goods”, Appl. Soft Comput, 94, 106462.

Bentner, J., Bauer, G., Obermair, G.M, Morgenstern, I., and Schneider, J. (2001). “Optimization of the Time-Dependent Traveling Salesman Problem with Monte Carlo Methods”, Physical Review E, 64(3):036701.

Ezzatollah, A., Sobhan, J., Hannan, A.M., and Jose, A.G. (2022). “A Novel Travel Time Estimation Model for Modeling a Green Time-Dependent Vehicle Routing Problem in Food Supply Chain”, MDPI Sustainability, from: <https://doi.org/10.3390/su14148633>.

Kok, A.L., Hans, E.W., and Schutten, J.M.J. (2012). “Vehicle Routing Under Time-Dependent Travel Time: The Impact of Congestion Avoidance”, Computers and Operations Research, Elsevier, from: doi:10.1016/j.cor.2011.05.027.

Marlin, U., Justin, C.G., Marco, H., and Dirk, M. (2018). “Offline-Online Approximate Dynamic Programing for Dynamic Vehicle Routing with Stochastic Requests “, from: <https://doi.org/10.1287/trsc.2017.0767>.

Orda, A., and Rom, R. (1990). “Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length”, Journal of the Association for Computing Machinery, 37(3), pp. 607–625.

Schneider, J. (2002). “The Time-Dependent Traveling Salesman Problem”, Physica A: Statistical Mechanics and its Applications, Elsevier, 314(1–4):151–155.

Van-Woensel, T., Kerbache, L., Peremans, H., and Vandaele, N. (2008). “Vehicle Routing with Dynamic Travel Times: a queueing approach”, European Journal of Operational Research, 186(3), pp. 990–1007.

**Chapter Two**

**Literature Review**

* 1. **Introduction**

Vehicle routing has become a problem of global interest to the researchers and business communities due to its application in logistics and transport services. Vehicle routing problem (VRP) is notably a subset of path planning problems.

Path planning problem revolves round the techniques of guiding a moving object or an agent through a collision free path in a specific environment. (Wan-Ngah et.al, 2010) defined path planning as the determination of a path that a robot must take to pass over each point in an environment and path is a plan of geometric locus of the points in each space where the robot has to pass through. It is usually considered in optimization, combinatorial and swarm coordination problems.

(Stentz, 1994) affirmed that the objective of a path planner is to proffer a suitable and proficient technique that can move an object (probably a robotic agent) from a location to a destination in a given environment, such that it avoids all obstacles and minimizes the cost to reaching the destination. The problem space can be considered as a set of states or locations denoting robot or vehicular area of free movement. The robot or vehicle starts at a particular state and moves across arcs to other states until it reaches its destination.

(Braekers et.al, 2015) reported that modern VRP solutions have evolved from the solution proposed by Dantzig and Ramser in 1959; they put into perspective real life considerations and realities. The advent of modern computers with high speed and memory capacity have aided several ingenuine implementations of VRP solutions with the possibility to explore all possible routing scenarios. (Eksioglu et.al, 2009) confirmed that research contribution to VRP solutions have been growing at the rate of 6% each year. More so, VRP is classified as a NP-hard problem; such problems have no guarantee that the possible solutions will always be optimal.

Some of the common vehicle routing problems or scenarios; (Braekers et.al, 2015) reported a classical VRP also dubbed as Capacitated VRP, (Montoya-Torres et.al, 2015) reported a survey on Multi Depot VRP (MDVRP), (Huang et al, 2017; Ben-Ticha et al, 2019; Gmira et al, 2021; Kok et.al, 2012) reported on Time Dependent VRP (TDVRP), (Li et.al, 2015; Yu et.al, 2016; Tan et.al, 2019) reported on VRP with Pickup and Delivery (PDPVRP), and (Campbell & Wilson, 2014; Gulczynski et.al, 2011) reported on Periodic VRP (PVRP) which comes handy for planning deliveries to customers over a certain period of time.

Generally, VRPs can be classified as being static or dynamic. It is considered static if the inputs to the solutions are fixed such as fixed travel time, delay time, number of vehicles, number of deliveries per trip and capacities of vehicle, but in a very dynamic world, the afore mentioned parameters are constantly changing due to market dynamics and other environmental factors beyond the control of the route planner. Consequently, the research community has developed several dynamic VRP models to accommodate the real time changes.

VRP is a combinatorial problem, and it requires optimization techniques to arrive at considerable solutions. Usually, VRP solutions are derived from heuristics, meta-heuristics, stochastics, dynamic programming, evolutionary and multidisciplinary methods.

* 1. **Variants of Vehicle Routing Problems**

There are several variants of VRP due to the versed routing scenarios and expectations. Very complex transport and operational requirements are constantly throwing up new challenges in VRP. More so, businesses want to surpass minimum service level agreement hence the higher demands to optimize critical factors that determine vehicle routing solutions. For instance, food and perishable goods industries want to deliver faster than ever, and public taxi services want to proffer alternative route real time to riders. (Montoya-Torres et.al, 2015; Braekers et.al, 2016) reported extensively on the variants of VRP.

* + 1. **Capacitated VRP (CVRP)**

CVRP is vehicle routing problem that involves capacity-based constraints such as number of trips per day, number of items that can be delivered per trip, number of available vehicles etc. (Akhand et.al, 2017) described CVRP as a basic form of VRP such that all vehicles have similar characteristics, depart from a single location and must return to the same location on completion of task. The objective of CVRP is to minimize the total traveling cost for the dispatchers. (Glock and Grosse, 2012; Scholz et al., 2017) proposed a variant of CVRP with capacity constraint based on batching of orders for deliveries that must be completed under a few trips. (Nurcahyo et.al,2002) proposed a sweep based VRP for public transport system in Indonesia. (Bruglieri et.al, 2017) proposed a CVRP for selecting alternative fueling stations. (Faulin et.al, 2011) proposed a CVRP solution with dynamic environmental factors as input. (Froger et.al, 2017) proposed a metaheuristic CVRP solution for capacitated charging station scenario. (Hof et.al., 2017; Jie et.al., 2019) worked on a CVRP solution that can manage battery swap arrangement for capacitated electric vehicles. (Lysgaard, 2010) proposed a pyramidal road network for CVRP such that the dispatchers start by dropping goods in increasing order of the predefined customer index and the remainder in a decreasing order of the customer index.

(Braekers, 2015) reported several VRP solutions that hinge on capacity such as capacitated Time Dependent VRP, capacitated Multi Depot VRP and capacitated Green VRP and capacitated Pickup and Delivery.

* + 1. **Time Dependent VRP (TDVRP)**

TDVRP is a VRP solution that takes cognizance of traffic congestion and peak period while designing an optimal solution. Apart from distance and other determinant factors time criticality is most important. (Malandraki and Daskin,1992) first proposed a time dependent VRP solution as a travel time step function for customer locations. (Malandraki and Dial, 1996) also proposed a solution for Time Dependent Travel Salesman problem which involves single vehicle or robot completing the routing task. (Kok et.al, 2012) noted that TDVRP depends on location and time of the day. (Jung and Haghani, 2005) proposed a genetic algorithm solution to VRP and validates the performance with an alternative solution using customer base of thirty (30). (Donati et.al, 2008) implemented a solution based on multi ant colony system to solve VRP. (Van-Woensel et.al, 2008) implemented a queuing model to solve VRP; the times were put in queue and pushed in certain order. (Braekers et.al, 2015) affirmed that TDVRP have several real-life applications as many human activities usually have time constraint.

* + 1. **Multi Depot VRP (MDVRP)**

Multi-depot VRP is a VRP scenario where delivery vehicles have may pick up items from different locations. (Narasimha et.al, 2018) proposed a solution for multi-depot capacitated VRP; this solution addresses vehicles that have a target or could be assigned task randomly to deliver items to customers maybe based on proximity or urgency. (Nazari et.al, 2018) employed greedy and beam search selection schemes and enforced the following constraints: nodes with no customer request are avoided, nodes with served customers will be excluded from subsequent selection, customers with higher demand than the remaining goods are postponed to next trip, more importantly, the vehicle must meet all the demand of a customer if it visited the location. MDVRP is sometimes matched with CVRP and other types of VRP. (Xu et.al, 2018) implemented ant colony optimization for dynamic multi-depot VRP (DMDVRP). A DMDVRP is a variant of MDVRP where constraints are not fixed but could assume different values during routing.

* + 1. **Green VRP (GVRP)**

(Iwata and Matsumoto, 2016) reported that due to the advancement in technology, research contributions have increased significantly towards reducing greenhouse effects and emissions, thereby opening up new sources of energy that are eco-friendly and usually renewable. (Asghari et.al, 2020) reported that (Erdogan and Miller-Hooks, 2012) were pioneers of green VRP and since 2010 VRP solutions that focus on pollution emission and energy conservation have skyrocketed. GVRP is also known as green logistics. Research works in GVRP tend to reduce energy utilization in fleet management, adoption of alternative-fuel powered vehicles (AFVs) or with known internal combustion engines vehicles (ICEVs) in fleets. (Xiao et.al, 2012) noted that if fewer fossil fuel was used, there will significant reduction in greenhouse gases effects and overall environmental pollution. (Kuo, 2010; Zeng et al., 2020) affirmed that fossil fuel reduction is one of the critical factors in designing a viable solution for GVRP. (Xiao et al, 2020) extended the discrete Pollution Routing Problem (a subset of GVRP) proposed by (Bektas and Laporte, 2011) to manage a continuous scenario by using the travel speed as a continuous decision variable. This ensures that all related scheduling factors such as load flow, travel time, departing or arrival or waiting times are all considered as continuous decision variables.

Pollution

Transportation

Fuel/Energy

Consumption

Figure 1: Classification of GVRP research Areas, adapted from (Asghari et.al, 2020)

* + 1. **Pickup and Delivery VRP (PDVRP)**

PDVRP is a vehicle routing problem that focuses on the task of delivering items at some locations and picking up some items from one or more locations during the trip. List of customers expecting deliveries are known as Linehauls while list of customers expecting pickups are called Backhauls. PDVRP have been hybridized with other forms of VRPs; such as capacitated PDVRP (CPDVRP) and Multi-Depot PDVRP (MDPDVRP). (Berbeglia et.al, 2007) did a comprehensive compilation of PDVRP solutions and challenges. (Li et.al, 2021) proposed a learning algorithm to solve VRP for multi pickup and delivery points. (Braekers et.al, 2015) mentioned that originally pickup and delivery was done by the same vehicle but (Lahyani et.al, 2015) proposed more realistic scenario to PDVRP where in multi-depot or pickup arrangement different vehicles could be assigned to do pickup.

* 1. **Optimization Algorithms Used in VRP**

VRP is classified to be a subset of the family of path planning problems. Notably, this class of problems are optimization problem. The planner or driver is interested in finding the most optimal conditions that meets its interest; it could be finding the shortest possible distance to a destination, shortest possible arrival or delivery time, best possible delivery service sequence or road networks that have the highest accumulation of green time. For various economic or research interest, different optimization algorithms and techniques have been explored to see the results on different VRP scenarios.

* + 1. **Ant Colony Optimization (ACO)**

It is a meta-heuristic optimization algorithm that is inspired from the cooperative and foraging behaviours observed in ant colonies. It was first proposed by Marco Dorigo in 1992 in his PhD thesis. From that time onward, ACO has become very popular and has been applied in varying optimization scenarios in different works of life. (Othman et.al, 2018) implemented an ACO solution for VRP problem. (Xu et.al, 2018) implemented a hybridized ACO solution to solve a multi-depot vehicle routing problem in record time. (Crispim and Brando, 2005) proposed an ACO solution to VRP scenarios with backhauls.

(Blum, 2005) affirmed that ACO was inspired by the ants’ foraging behavior. At the core of this behavior is their indirect mode of communication. They encode messages in their environment by laying a trail of chemical pheromone, which enables them to find short paths between their nest and food sources. This characteristic of real ant colonies is exploited in ACO algorithms.

The solution is modeled as follow: the chemical pheromone is represented as τ (this represents the concentration of the pheromone on any given trail), n number of ants (this represents possible solutions), attractive, evaporation and other dependent parameters.

Each ant starts from the nest and chooses the next node with the probability value (*P)*

i= 1, 2……n

The pheromone (*T*) is assumed to evaporate after a while during the foraging activity, which helps the oncoming ants identify less reinforced paths and follow paths with higher *T* values. The formulas below represent Pheromone concentration and update respectively.

(2)

(3)

Given a source and destination points, ACO can search for the shortest possible distance between the two points in quality time.

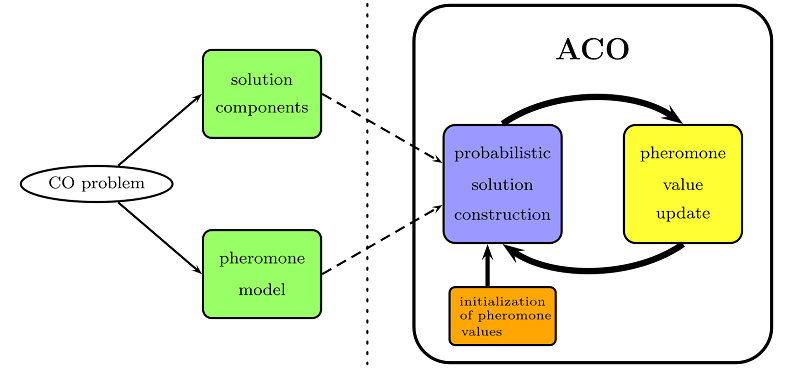


Figure 2: Adapted from Blum (2005): ACO Metaheuristic framework.

|  |
| --- |
| **Ant colony optimization (ACO)** |
| Determine number of ants, cities and other parameters  Initialize Ants trails  Initialize Pheromones for trails  **While** (Condition not True) **Execute**  ConstructSolution ( )  UpdatePheromone ( )  **End While** |

Figure 3: Basic Ant System Algorithm

There are several variants of ACO that have been applied to solve varying optimization problems.

|  |  |
| --- | --- |
| **ACO Variants** | **Authours** |
| Elitist AS (EAS) | Dorigo, Maniezzo, and Colorni |
| Rank-based AS (RAS) | Bullnheimer, Hartl, and Strauss |
| Max-Min Ant System (MMAS) | Stutzle and Hoos |
| Ant Colony System (ACS) | Dorigo and Gambardella |
| Hyper Cube Framework (HCF) | Dorigo and Blum |

Table 1: A list of ACO variants. Adapted from Blum (2005)

* + 1. **Dijkstra Algorithm**

(Misa, 2010) reported that Edger W. Dijkstra designed a shortest path algorithm to solve the problem of moving along the shortest possible distances between Rotterdam and Groningen. This algorithm is useful for planning the shortest paths between nodes in a graph or network, which may represent, road, pipeline or communication networks. The algorithm may represent the nodes of the graph as cities, junctions with associated path costs. Dijkstra algorithm finds the shortest route between one city and all other cities. For a given source node in the graph, the algorithm finds the shortest path between that node and every other, producing a shortest path tree.

(Moussa, 2021) implemented a hybrid of recursive K-Means and Dijkstra algorithms to solve capacitated vehicle routing problem. (Odumosu et.al, 2018) adopted Dijkstra algorithm to plan less congested route for students in a GIS enabled environment. (Adna et.al, 2020) also implemented a Dijkstra algorithm to optimize solution for multi delivery or pickup problem in VRP. (Nha et.al, 2012) proposed an improved Dijkstra algorithm to simulate a solution for VRP in a smart city.

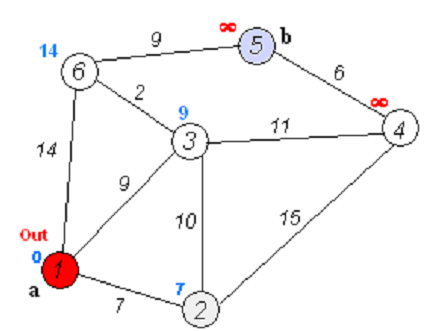


Figure 5: Sample Network, adapted from: Wikipedia

* + 1. **A Star (A\*)**

A\* is a popular heuristic search algorithms that has been extensively used for several path planning problems. (Misa, 2010) reported that A\* is a generalized form of Dijkstra shortest path algorithm. It is considered as one of the informed search techniques that extends the Best First search (BFS) technique, such that asides the weight cost along the edges it computes a heuristic function that further checks for the possibility of selecting a better option on every attempt while traversing the network. Peter Hart, Nils Nilsson and Bertram Raphael were first to describe this algorithm in 1968.

Given a graph, A\* traverses the graph and build up partial tree paths. The nodes of the tree form an open set (fringe) and are stored on a priority queue that assigns the leaf nodes based on the cost function, which combines a heuristic estimate of the cost to reach a goal and the distance traveled from the initial node. Specifically, the cost function is represented below

(4)

Function *g(n)* represents the known cost of getting from the initial node to a particular node n. Function *h(n)* is a heuristic estimate of the cost to get from a particular node n to the specified sub or global goal node. To find the actual shortest path, it is important the heuristic function does not overestimate the actual cost to reaching the nearest goal node. The heuristic function is problem-specific and must be provided by the user of the algorithm. Generally, heuristics are domain specific knowledge about the problem to be solved.

The A\* algorithm also has real-world applications in many systems; such as communication, path planning and generally in transportation. (Masari, 2019; Donne & Tagliavini, 2019) implemented A\* algorithm for dynamic VRP.

* + 1. **Learning Realtime A\* (LRTA\*)**

The first step towards making conventional heuristic search technique real time was LRTA\*. It precedes every other real time heuristic search algorithms. It is similar to learning techniques that adopt common temporal difference control theory models e.g., Q-learning. A vehicular agent starts by initializing its current location (*s*) as the starting position, then it generates a set of successor states around *s*’ neighborhood. It does an expansion of the local search space (network), its frontiers (possible connecting routes) states are considered the viable states to transit to. The chosen state is that which minimizes the objective function

(5)

Where is the cost of an optimal path from s to s’ within the local search space and is the heuristic cost from the potential next state to global goal. To avoid state re-visitation, the heuristic value) is assigned the total cost value in equation 1 above. The agent then moves one step along the path towards the goal. These actions are carried out until the agent’s position matches the goal state. LRTA\* is only complete given an admissible heuristic.

(Huntley, and Bulitko, 2013) reported a major setback of LRTA\*; scrubbing. Scrubbing is an optimization problem where a selection policy or mechanism continues to return already visited states or options which may get the agent trapped. If scrubbing not fixed, it will make LRTA\* sub optimal and not viable for real time commercial applications.

* + 1. **Time Bounded A\* (TBA\*)**

(Bjornson et.al, 2009) confirmed that TBA\* does not require construction of sub-goal database for the entire search space. TBA\* is essentially a time-sliced A\*. It builds its open and closed lists from the start state in the same way as A\*.

(Huntley and Bulitko, 2013) compares TBA\* to conventional A\*: it does not wait for the goal state to be at the head of its open list. Instead, after a fixed number of expansions, a TBA\*-driven agent takes one step towards the most promising state on the open list, if while on its way to the most promising open-list state, such a state changes, the agent re-plans its path towards the new target. All the planning operations are time-sliced so that the resulting algorithm is real-time.

* + 1. **Support Vector Machine (SVM)**

(Vapnik, 2000) described SVM as a binary classifier that selects an optimal separating hyper plane based on margin maximization. SVM is widely used in data mining and machine learning.

(Miura, 2006) was first to adapt support vector machine to solving path planning problem. Obstacles in the search space are labelled as positive and negative samples. The problem is transformed into the dual problem of finding a smooth separating surface for the two classes. His approach was only applied to static or less turbulent environment.

(Sarka et.al, 2008; Tennety et.al, 2009) applied kernel-based support vector machine to plan a collision free path in a dynamic environment. (Tennety et.al, 2009) made the approach very reactive by using K-means which assist to classify the unknown obstacles while the agent is progressing towards some goal in the search space. The SVM classification performed very excellently in mapping out the collision free paths.

However, (Qingyang, 2012) reported some setbacks with the previous approaches, in that the kernel-based support vector machine does not guarantee the separation of intermediate region between the origin and goal. It also does not give precise collision free path or guarantee that the planned path being coincident with the direction of the vehicle. It also requires the boundary points of any route and uses obstacles samples to search for an optimal path for the training process.

(Qingyang, 2012) improved on the previous effort by extracting the topology of the search space and this helps to easily provide the requirement for the boundary points unlike the bottleneck in previous efforts that requires constant reference to obstacle samples and boundary point in every training process.

(Guan et.al, 2016) used SVM to solve VRP for shipping and delivery. (Guan et.al, 2019) used SVM to map historical and current routing data into two dimensional planes.

* + 1. **Learning Algorithms in VRP**

This class of solutions use artificial intelligence (AI) and related learning techniques in managing vehicular routing. The traffic factors to be considered must be interpreted and represented at the system level in line with the AI model or techniques.

(Bdeir et.al, 2021; Tamagawa et.al, 2010) implemented Q-Learning solution for VRP, (Chen et.al,2021) proposed a Q-Learning solution to doing deliveries with vehicles and drones. (Sedighizadeh, and Mazaheripour, 2018) proposed a solution based on particle swarm optimization to solve a multi objective vehicle routing problem. (Yu et.al, 2019) combined deep learning and other optimization algorithm to solve VRP. (Li et.al, 2021) proposed a solution by modeling the CVRP problem as MDP and adopting neural network to optimize the stochastic policy which should lead the best situation. (Li et.al, 2022; Nazari et.al, 2018; Kool et.al, 2018, Xin et.al, 2021) all applied some learning models to solving VRP with better results and in good time.

* 1. **Cloud Computing**

Cloud computing is the provision of computing services over the internet. Basically, all computing activities we do locally on our computers can be organized and made available over the internet. There are private and public cloud. Big organizations who have capacities to buy high end hardware and employ the services of technical teams to support their infrastructure build and operate computing services over a private internet connection to do their business operations. (Salmon and Parmer, 2022) defined cloud computing as interconnected servers that are used for providing computing services over the internet. (Jadeja and Modi, 2012) defined cloud computing as computing space where clients outsource their computing needs to cloud service providers and such services are provided on demand and at a shared cost. Usually, the underlying hardware configuration and management may not be visible to the client, the cloud providers ensure that upkeep of the computing environment in terms of power, licenses and technical know-how are sorted.

The advent of global pandemic has opened the cloud ecosystem with the ability to support business operations remotely and efficiently. Before the pandemic, the cloud business does less than $200 Billion globally but it brings in over $500 Billion. (Pearl et.al, 2022) reported the staggering growth in cloud technologies and how they have changed the landscape of computing services and daily business operations.

There are different computing models: Software as a service (Saas), Platform as a service (Paas) and Infrastructure as a service (Iaas). Saas are finished software products or services available in the cloud, but the users just are allowed to do minimum configuration and bring their data to the Saas’ platform. They do not have to bother about maintenance, platform-wide security, or technical know-how. Paas allows users to bring their data and application of choice but they are shielded from the background running of the hardware, networking and maintenance. Iaas allows the users to setup the computing environment to their taste; they have much control over the hardware than Saas and Paas. In Iaas, users would have to provide for technical know-how, operating system, networking and license. Unlike Iaas, users of Saas and Paas are made to focus more on the business operations rather than focusing on the hardware issues and its dependencies.

* 1. References

Adna, S., Abood, E.W., and Abdulmuhsin, W. (2020). “The Multi-point delivery Problem: Shortest Path Algorithm for Real Roads Network Using Dijkstra”, Journal of Physics, IOP Publishing, doi:10.1088/1742-6596/1530/1/012040

Akhand, M.A.H., Peya, Z.H., and Murase, K. (2017). “Capacitated vehicle routing problem solving using adaptive sweep and velocity tentative PSO”, International Journal of Advance Computer Science and Applications, 8(12), pp. 288-295.

Asghari, M., and Mirzapour Al-e-hashem, S.M.J. (2020). “Green Vehicle Routing Problem: A State-of-the-Art Review”, International Journal of Production Economics (2020), <https://doi.org/10.1016/j.ijpe.2020.107899>.

Bektaş, T., and Laporte, G. (2011). “The pollution-routing problem”, Transp. Res. B Methodol, 45, pp. 1232–1250.

Bdeir, A., Dernedde, T., Tkachuk, K., Falkner, J.K., and Schmidt-Thieme, L. (2021). “RP-DQN: An application of Q-Learning to vehicle routing problems”, arXiv:2104.12226v1.

Ben-Ticha, H., Absi, N., Feillet, D., Quilliot, A., and Van-Woensel, T. (2019). “A branch-and-price algorithm for the vehicle routing problem with time windows on a road network”, Networks 73 (4), pp. 401–417.

Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: A classification scheme and survey. TOP, 15(1), pp. 1–31, <http://doi.org/10.1007/s11750-007-0009-0>

Bjornsson, Y., Bulitko, V., and Sturtevant, N. (2009). “TBA\*: Time Bounded A\*” <https://www.ijcai.org/Proceedings/09/Papers/079.pdf>

Blum, C., and Dorigo, M. (2005). “ Ant Colony Optimization Theory: Survey”, Theoretical Computer Science, 344, pp. 243-278, <https://doi.org/10.1016/j.tcs.2005.05.020>.

Bruglieri, M., Mancini, S., Pezzella, F., and Pisacane, O. (2017a). “The green vehicle routing problem with capacitated alternative fuel stations”, Verolog 2017, Amsterdam, July 10th-2th.

Bruglieri, M., Mancini, S., Pezzella, F., and and Pisacane, O., (2019a). “A path-based solution approach for the green vehicle routing problem”, Computers & Operations Research. 103, pp. 109–122.

Bruglieri, M., Mancini, S., Pezzella, F., and Pisacane, O. (2019b). “The green vehicle routing problem with capacitated alternative fuel stations”, Computers & Operations Research. 112, pp. 104759

Chabot, T., Lahyani, R., Coelho, L. C., & Renaud, J. (2017). “Order picking problems under weight, fragility and category constraints”, International Journal of Production Research, 55(21), pp. 6361-6379.

Chen, X., Ulmer, M.W., and Thomas, B.W. (2021). “Deep Q-Learning for Same-Day Delivery with Vehicles and Drone”, European Journal of Operational Research, Elsevier. <https://doi.org/10.1016/j.ejor.2021.06.021>

Crispim, J., and Brand, A.O. (2005). “Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls”, Journal of the Operational Research Society, 56, pp. 1296–1302.

Donati, A.V., Montemanni, R., Casagrande, N., Rizzoli, A.E., and Gambardella, L.M. (2008). “Time dependent vehicle routing problem with a multi ant colony system”, European Journal of Operational Research 2008, 185(3), pp.1174–91.

Donne, D.D., and Taglavini, G. (2019). “Star Routing: Between Vehicle Routing and Vertex Corver”, arXiv:1901.09154v1.

Eksioglu, B., Vural, A. V., and Reisman, A. (2009). “The vehicle routing problem: A Taxonomic Review”, Computers & Industrial Engineering, 57(4), pp. 1472–1483, <http://doi.org/10.1016/j.cie.2009.05.009>.

Erdogan, S., and Miller-Hooks, E. (2012). “A green vehicle routing problem”, Transp. Res. E Logist. Transp. 48 (1), pp. 100–114.

Faulin, J., Juan, A., Lera, F., and Grasman, S. (2011). “Solving the capacitated vehicle routing problem with environmental criteria based on real estimations in road transportation: a case study”, Procedia - Social and Behavioral Sciences, 20, pp. 323–334.

Froger, A., Mendoza, J.E., Jabali, O., and Laporte, G. (2017a). “A metaheuristic for the electric vehicle routing problem with capacitated charging stations”, Technical report.

Campbell, A. M., & Wilson, J. H. (2014). “Forty Years of Periodic Vehicle Routing”, Networks, 63(1), pp. 2–15. <http://doi.org/10.1002/net.21527>.

Huang, Y., Zhao, L., Van-Woensel, T., and Gross, J.P. (2017). “Time-dependent vehicle routing problem with path flexibility”, Transp. Res. Part B 95, pp. 169–195.

Jadeja, Y., and Modi, K. (2012). “Cloud Computing- Concepts, Architecture and Challenges”, DOI:10.1109/ICCEET.2012.6203873

Glock, C.H., and Grosse, E.H. (2012). “Storage policies and order picking strategies in u-shaped order-picking systems with a movable base”, Int J Prod Res 50(16), pp.4344–4357

Gmira, M., Gendreau, M., Lodi, A., and Potvin, J.Y. (2021). “Tabu search for the time-dependent vehicle routing problem with time windows on a road network”, European J. Oper. Res. 288 (1), pp. 129–140.

Guan, F., Peng, Z.X., and Wang, K.M. (2016). “Multi-step hybrid prediction model of Baltic Supermax Index based on support vector machine”, Neural Netw World 2016, 26: pp. 219–232.

Guan, F., Shen, W., Wu, L., Yu, Y., Sun, D., and Yang, Y. (2019). “Fleet Route Selection Prediction Problem Based On Support Vector Machine”, Data Driven Optimization-Research Article: Advances in Mechanical Engineering, <https://doi.org/10.1177/1687814019836857>.

Gulczynski, D., Golden, B., and Wasil, E. (2011b). “The period vehicle routing problem: New heuristics and real-world variants”, Transportation Research Part E: Logistics and Transportation Review, 47(5), pp. 648–668. <http://doi.org/10.1016/j.tre.2011.02.002>

Haghani, A., and Jung, S.A. (2008). “Dynamic Vehicle Routing Problem With Time-Dependent Travel Times”, Computers & Operations Research 2005, 32(11), pp. 2959–86.

Hof, J., Schneider, M., and Goeke, D. (2017). “Solving the battery swap station locationrouting problem with capacitated electric vehicles using an AVNS algorithm for vehicle-routing problems with intermediate stops”, Transp.Res. B Methodol, 97, pp. 102– 112.

Huntley, D., and Bulitko, V. (2013). “Search Space Characterization for Real Time Heuristic Search”, arXiv:1308.3309v1

Iwata, K., and Matsumoto, S. (2016). “Use of hybrid vehicles in japan: an analysis of used car market data”, Transp. Res. D Transp. Environ, 46, pp. 200–206.

Jie, W., Yang, J., Zhang, M., and Huang, Y. (2019). “The two-echelon capacitated electric vehicle routing problem with battery swapping stations: formulation and efficient methodology”, European Journal of Operational Research. 272 (3), pp. 879–904.

Kool, W., Hoof, H.V., and Welling, M. (2018). “Attention, Learn to Solve Routing Problems!” in International Conference on Learning Representations.

Kuo, Y., 2010. “Using simulated annealing to minimize fuel consumption for the timedependent vehicle routing problem. Computers & Industrial Engineering. 59(1), pp. 157– 165.

Li, J., Xin, L., Cao, Z., Lim, A., Song, W., and Zhang, J. (2021), “Heterogeneous Attentions for Solving Pickup and Delivery Problem Via Deep Reinforcement Learning,” IEEE Transactions on Intelligent Transportation Systems.

Li, J., Ma, Y., Gao, R., Cao, Z., Lim, A., Song, W., and Zhang, J. (2022). “Deep Reinforcement Learning for Solving the Heterogenous Capacitated Vehicle Routing Problem”, IEEE Transaction on Cybernetics, arXiv:2110.02629v2.

Li, H., Lv, T., and Li, Y. (2015). “The tractor and semitrailer routing problem with many-to-many demand considering carbon dioxide emissions”, Transp. Res. D Transp. Environ. 34, pp. 68-82.

Lysgaard, J. (2010). “The pyramidal capacitated vehicle routing problem”, European Journal of Operational Research, 205(1), pp. 59–64. <http://doi.org/10.1016/j.ejor.2009.11.029>

Malandraki, C., and Daskin, M.S. (1992). “Time dependent vehicle routing problems: formulations, properties and heuristic algorithms”, Transportation Science 1992, 26(3), pp. 185–200.

Malandraki, C., and Dial, R.B. (1996). “A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem”, European Journal of Operational Research 1996, 90(1), pp. 45–55

Masari, I. (2019). “ Vehicle Routing Optimization Algorithm Based on A-Star (A\*)”, Research gate.

Misa, T. (2010). An Interview with Edsger W. Dijkstra. Communications of the ACM 53 (8), pp. 41–47, 10.1145/1787234.1787249.

Miura, J. (2006). Support vector path planning. IEEE International Conference on Intelligent Robots and Systems, Beijing, China pp. 2894‐2899.

Montoya-Torres, J.R., Franco, J.L., Isaza, S.N., Jimenez, H.F., and Herazo-Padilla, N. (2015). “A Literature Review on the Vehicle Routing Problem with Multiple Depots”, Computer and Industrial Engineering, 79 (2015), pp. 115-129, <http://dx.doi.org/10.1016/j.cie.2014.10.029>.

Moussa, H. (2021). “Using Recursive K-Means and Dijkstra Algorithm to Solve CVRP”, DOI:10.13140/RG.2.2.20970.85447

Narasimha, K.V., Kivelevitch, E., Sharma, B., and Kumar, M. (2013). “An Ant Colony Optimization Technique for Solving Min–Max Multi-Depot Vehicle Routing Problem,” Swarm and Evolutionary Computation, vol. 13, pp. 63–73.

Nazari, M., Oroojlooy, A., Snyder, L., and Takac, M. (2018). “Reinforcement Learning for Solving the Vehicle Routing Problem,” in Advances in Neural Information Processing Systems, pp. 9839–9849.

Nha, V.T.N., Djahel, S., and Murphy, J. (2012). “A comparative Study of Vehicles’ Routing Algorithms for route planning in smart cities”, 2012 First International Workshop on Vehicular Traffic Management for Smart Cities, IEEE. <https://doi.org/10.1109/VTM.2012.6398701>

Nurcahyo, G. W., Alias, R. A., Shamsuddin, S. M., and Sap, M. N. (2002). “Sweep algorithm in vehicle routing problem for public transport,” Journal Antarabangsa (Teknologi Maklumat), vol. 2, pp. 51-64.

Othman, W.A.F., Wahab, A.A A., Alhady, S.S.N., and Wong, H.N. (2018). “Solving vehicle routing problem using ant colony optimization (ACO) algorithm”, International Journal of research and engineering. <http://dx.doi.org/10.21276/ijre.2018.5.9.2>

Odumosu, J.O., Morenikeji, O.O., and Adesina, E.A. (2018). “ Vehicle route optimization for student transportation in a GIS and programming environment: Case Study of Mawo Secondary School, Minna, Niger State, Nigeria. DOI: 10.4314/njtr.v13i2.15

Pearl, M., Christy, S., Hollis, C., and Johnston, D. (2021). “Cloud Computing: Global Overview”, 2021 Law Business Research Ltd, Lexology.

Qingyang, C., Zhenping, S., Daxue, L., Yuqiang, F., and Xiaohui. (2012). “Local Path Planning for an Unmanned Ground Vehicle Based on SVM”, International Journal of Advanced Robotic Systems. Int J Adv Robotics Sys, Volume. 9,246: [www.intechopen.com](http://www.intechopen.com)

Salmon, M., and Parmer, A. (2022). “Cloud Computing at Unitec”, DOI:10.13140/RG.2.2.16483.63528

Sarkar, S., Hall, E., and Kumar, M. (2008). “Mobile robot path planning using support vector machines”. ASME Dynamic Systems and Control Conference, October 20-22, Ann Arbor, Michigan, USA.

Scholz, A., Schubert, D., and Wäscher, G. (2017). “Order picking with multiple pickers and due dates– Simultaneous solution of Order Batching, Batch Assignment and Sequencing, and Picker Routing Problems”, European Journal of Operational Research, 263(2), pp. 461-478.

Sedighizadeh, D., and Mazaheripour, H. (2018). “Optimization of Multi Objective Vehicle Routing Problem Using a New Hy-brid Algorithm Based on Particle Swarm Optimization and Artificial Bee Colony Algorithm Considering Precedence Constraints”, Alexandria Engineering Journal, 2018, 57(4), pp. 2225-2239. https://doi. org/10.1016/j.aej.2017.09.006

Stentz, A. (1994). “Optimal and Efficient Path Planning for Partially Known Environments”, In Proceedings IEEE International Conference on Robotics and Automation.

Tamagawa, D., Taniguchi, E., and Yamada, T. (2010). “Evaluating City Logistics Measures Using a Multi-Agent Model”, The Sixth International Conference on City Logistics, doi:10.1016/j.sbspro.2010.04.014

Tan, Y., Deng, L., Li, L., and Yuan, F. (2019). “The capacitated pollution routing problem with pickup and delivery in the last mile”, Asia Pacific Journal of Marketing and Logistics, 31 (4), pp. 1193-1215.

Tennety, S., Sarkar, S., Hall, E., Kumar, M. (2009). “Support vector machines based mobile robot path planning in an unknown environment”, Proceedings ‐ ASME  Dynamic  Systems  and  Control  Conference,

Vapnik, V. (2000) The Nature of Statistical Learning Theory Second Edition. New York: Springer‐Verlag.

Van-Woensel, T., Kerbache, L., Peremans, H., and Vandaele, N. (2008). “Vehicle routing with dynamic travel times: a queueing approach”, European Journal of Operational Research 2008;186(3): pp. 990–1007.

Wang, L.C., Yong, L.S., and Ang M.H. (2002). “Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment”, Computer Science Proceeding of the IEEE International Symposium on Intelligent Control.DOI:10.1109/ISIC.2002.1157868

Wan-Ngah, W.A.J., Buniyamin, N., and Mohamad, Z. (2010). “Point to Point Sensor Based Path Planning Algorithm for Mobile Robots”, In 9th WSEAS International Conference on System Science and Simulation in Engineering (ICOSSSE'10). Iwate, Japan.

Xiao, Y., Zhao, Q., Kaku, I., Xu, Y. (2012). “Development of a fuel consumption optimization model for the capacitated vehicle routing problem”, Computers & Operations Research 39 (7), 1419–1431.

Xin, L., Song, W., Cao, Z., and Zhang, J. (2021). “Multi-Decoder Attention Model with Embedding Glimpse for Solving Vehicle Routing Problems”, in Proceedings of 35th AAAI Conference on Artificial Intelligence.

Xu, H., Pu, P., and Duan, F. (2018). “A Hybrid Ant Colony Optimization for Dynamic Multidepot Vehicle Routing Problem”, Discrete Dynamics in Nature and Society. <https://doi.org/10.1155/2018/3624728>

Yu, J.J.Q., Yu, W., and Gu, J.T. (2019). “Online Vehicle Routing with Neural Combinatorial Optimization and Deep Reinforcement Learning”, IEEE Transactions on Intelligent Transportation Systems, 2019, 20(10), pp. 3806-3817. <https://doi.org/10.1109/TITS.2019.2909109>

Yu, Y., Tang, J., Li, J., Sun, W., and Wang, J. (2016). “Reducing carbon emission of pickup and delivery using integrated scheduling”, Transp. Res. D Transp. Environ. 47, pp. 237– 250.

Zeng, W., Miwa, T., Morikawa, T. (2020). “Eco-routing problem considering fuel consumption and probabilistic travel time budget”, Transp. Res. D Transp. Environ, 78, 102219.

**Chapter Three**

**Research Methodology**

* 1. **Problem description**

Design the proposed road network as a graph with connecting nodes and edges as road, generate traffic values such as average flow rate along each road, model a solution using Markov Decision Process and improved Q-Learning algorithm to select the best route from one point in the network to another point in the network. Deploy the solution to a public cloud using Azure Machine Learning Studio.

In summary the problem is to develop a strategy or model that suggests the most efficient route from one point to another point within the proposed road network with possible traffic constraints.

* 1. **Methodology**

This section explains the approach to solving the problem described in 3.1 above.

* + 1. **Road Network Topology**

This section depicts the sample road network that is being understudied. The sample network is adopted as a model of the actual road network. Usually, a network comprises of node points, edges or lines connecting the nodes. The edges are not always of the same length and there could be constraints in terms movement. For simplicity this road network will assume node tags from A to K which represent junctions while the edges will represent the actual roads or paths connecting the junctions. Section 3.2.3 below depicts the movement of freedom among the nodes such that if there is a connection from node A to C, the edge or road will be tagged as AC and the intersection on the table will have a value 1, while blank spaces on the table indicate zero degree of freedom from the current node to the intersecting node.

I

B

J

F

D

K

H

A

E

C

G

Figure 6: Proposed Road Network

* + 1. **Markov Decision Process (MDP)**

The proposed solution is modeled as a control theory problem using Markov decision process (MDP). MDP is one of the many combinatorial design models. It is a feedback compliant system that ensures every action is rewarded and the reward helps the learning agent to take smarter decisions as the learning activity progresses. (Yue and Jesus, 2022) acknowledge MDP as the mathematical building block for reinforcement learning (RL) and has brough great results in combinatorial based problems. MDP is usually defined as a 4-tuple (*S, A, P, R*).

where:

***S***= *{S*1*, S*2*, ..., Sn}* is a finite set of states.

***A***= *{A*1*, A2, ..., Am}* is a finite set of actions.

***P***is a Markovian state transition model or action-state mapping function — ***P*** (*S, A, S\_*) is the probability of making a transition from state S to *S\_* by taking action ***A****,* ***P*** (*S −A→ S\_*) and,

***R***is a reward (or cost) function — ***R*** (*S, A, S\_*) is the reward for the transition *S −A→ S\_*.

The nodes are assumed to be the set of possible states S; the set of connecting nodes alongside their respective average flow rate constitute the possible set of action A, from any given node. The model also has a reward scheme that helps the learning agent understands if it is doing well. These four MDP components are sufficient to describe the system and serve as moderating factors for the learning algorithm that is used to optimize the MDP decisions.

Hence, the description of the components of the MDP in relation to the proposed system is as follows:

* + - 1. **States**

Based on the network topology in section 3.21 above, MDP represents the nodes as states

|  |  |  |
| --- | --- | --- |
| **S/N** | **Nodes** | **Connection** |
| 1 | A | AC, AB, AE |
| 2 | B | BA, BD, BF |
| 3 | C | CA, CE, CG |
| 4 | D | DB, DE |
| 5 | E | EA, EC, ED, EF, EG, EH |
| 6 | F | FB, FE, FI, FH |
| 7 | G | GC, GE, GK |
| 8 | H | HE, HJ |
| 9 | I | IF, IJ |
| 10 | J | JH, JI, JK |
| 11 | K | KG, KJ |
|  |  |  |

Table 2: State Representation in MDP

* + - 1. **Actions**

MDP expects certain actions that will bring about transition from one state to another. It is also expected that when an action is taken there should be some reward. The reward serves as feedback that helps to moderate the performance of the learning agent towards reaching its destination with an optimal value. As represented in the table below; a learning agent or vehicle on the road can perform ***forward movement*** towards a new road or ***backward movement*** to a previously visited road.

|  |  |  |  |
| --- | --- | --- | --- |
| S/N | Action | Reward | States |
| 1 | Forward Movement | 10 | Any state in S |
| 2 | Backward Movement | -10 | Any state in S |

Table 3: Action-Reward Scheme

* + - 1. **State-Transition Matrix**

This table depicts the degree of freedom among the states from any given state. The columns and headers form intersections such that the connection between any two states is represented as 1 and states that do not have connection is marked as dash (-).

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** |
| **A** | - | 1 | 1 | - | 1 | - | - | - | - | - | - |
| **B** | 1 | - | - | 1 | - | 1 | - | - | - | - | - |
| **C** | 1 | - | - | - | 1 | - | 1 | - | - | - | - |
| **D** | - | 1 | - | - | 1 | - | - | - | - | - | - |
| **E** | 1 | - | 1 | 1 | - | 1 | 1 | 1 | - | - | - |
| **F** | - | 1 | - | - | 1 | - | - | 1 | 1 | - | - |
| **G** | - | - | 1 | - | 1 | - | - | - | - | - | 1 |
| **H** | - | - | - | - | 1 | - | - | - | - | 1 | - |
| **I** | - | - | - | - | - | 1 | - | - | - | 1 | - |
| **J** | - | - | - | - | - | - | - | 1 | 1 | - | 1 |
| **K** | - | - | - | - | - | - | 1 | - | - | 1 | - |
|  |  |  |  |  |  |  |  |  |  |  |  |

Table 4: State Transition Mapping

* + 1. **Q-Learning Algorithm**

Q-Learning is a model-free reinforcement learning algorithm that can learn and predict the value of an action in any given state. A finite MDP Q-Learning tries to select an optimal policy by maximizing the expected value of the accruable reward. The agent starts out with a clean sheet by initializing the Q-Table to zero. The agent is expected to learn through experience, by exploring transition options of moving from state to state until it reaches its destination.

Equation 1 below is the general Q-Learning formula for computing a Q-value while equation 2 is an improved formula that takes into consideration the average flow speed on a road.

(6)

(7)

Where:

*α = Learning rate*

*S = Current state*

*A = Action taken*

*R = Reward received for taking action A.*

*Qo = Old Q-value on the Q-Table*

*Qn = New Q-value*

*AFR = Average Flow Rate on a road.*

* + - 1. **Q-Learning Procedure**
* Initialize Q-Table values for all state S to zero
* Do Policy Iteration (see section 3.2.3.2)
* Get Reward and Move to New State
* Do Q-Value Iteration (see section 3.2.3)
* Repeat previous steps until optimal value is reached i.e., arrived at the destination.
  + - 1. **Learning Policy**

The policy is a rule that defines how a decision should be made. The recurrence formula below describes the learning policy for the vehicular agent. The policy prescribes the suitable conditions that will lead to an optimal value by selecting next states by using a greedy scheme.

(8)

* + - 1. **Q-Table**

This is the data structure used by the algorithm to track how the learning agent is doing during the journey. The table is initialized to zero at the beginning and updated as the agent or vehicle takes action and gets rewarded. This will lead to computing new Q-values that will update the old values. This happens each time the learning agent moves from one state to another state. Q-Table depicts the brain of the learning agent.  The rows represent the current state while and columns represent the possible actions leading to the next state.

|  |  |  |
| --- | --- | --- |
|  | **Forward Movement** | **Backward Movement** |
| **A** | 0 | 0 |
| **B** | 0 | 0 |
| **C** | 0 | 0 |
| **D** | 0 | 0 |
| **E** | 0 | 0 |
| **F** | 0 | 0 |
| **G** | 0 | 0 |
| **H** | 0 | 0 |
| **I** | 0 | 0 |
| **J** | 0 | 0 |
| **K** | 0 | 0 |

Table 5: Initialized Q-Table

* + - 1. Average Flow Rate (AFR)

AFR is the average number of vehicles on the road within a period. Usually, different road will have different volume of vehicles at different time of the day. The intention is to use this information as part of the learning experience for the agent or vehicle. AFR below is captured on a three-hour basis i.e. (180 minutes in total time considered), Therefore equation 9 above, computes the average number of vehicles.

AFR (9)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Rate** | **9:00**  **-**  **12:00**  **pm** | **12:01**  **-**  **03:01**  **pm** | **03:02**  **-**  **06:02**  **pm** | **06:03**  **-**  **09:03**  **pm** | **09:04**  **-**  **12:04**  **am** | **12:05**  **-**  **03:05**  **am** | **03:06**  **-**  **06:06**  **am** | **06:07**  **-**  **09:0**  **am** |
| **A** | 5.00 | 3.34 | 1.26 | 3.80 | 1.25 | 1.00 | 1.20 | 1.40 |
| **B** | 4.52 | 5.54 | 6.52 | 3.82 | 2.52 | 0.52 | 1.52 | 4.52 |
| **C** | 3.48 | 3.82 | 4.48 | 3.98 | 1.48 | 0.48 | 1.28 | 3.48 |
| **D** | 4.40 | 2.40 | 5.40 | 4.00 | 2.40 | 0.40 | 1.40 | 4.40 |
| **E** | 2.90 | 4.90 | 3.90 | 3.20 | 1.90 | 0.90 | 1.90 | 2.90 |
| **F** | 3.20 | 4.34 | 4.20 | 3.80 | 1.20 | 0.20 | 1.20 | 3.20 |
| **G** | 2.80 | 3.92 | 3.80 | 3.80 | 2.80 | 0.80 | 1.40 | 2.80 |
| **H** | 4.60 | 3.96 | 5.60 | 3.60 | 2.60 | 0.60 | 1.60 | 4.60 |
| **I** | 3.90 | 4.30 | 4.90 | 2.90 | 1.90 | 0.50 | 1.50 | 3.90 |
| **J** | 2.20 | 4.10 | 3.20 | 2.80 | 1.20 | 0.20 | 1.20 | 2.20 |
| **K** | 2.80 | 2.90 | 3.80 | 2.90 | 1.80 | 0.60 | 1.60 | 2.80 |

Table 6: Average Flow Rate

The table above divides a day(24hrs) into eight cycles, 3hrs each. The numbers were randomly generated and could be replaced with a field data if available.

* 1. **Systems Requirement**

This section covers the tools and technology alongside required specifications that are needed to implement this solution.

* + 1. **Azure Machine Learning Studio**

Microsoft is one of the major players in the tech space. As part of her desire to register her foothold in artificial intelligence (AI) she has and continue to invest in AI inspired products and processes. Microsoft is a leading research organization in Augmented and Virtual realities (AR/VR), automated security, intrusion detection, signal processing, and automation of AI and machine learning product development. Microsoft has developed a flexible but systematic methodology through her platform, Azure machine learning studio.

Azure Machine Learning Studio is a GUI based integrated development environment used for developing and operationalizing AI and Machine Learning (ML) workflows on Microsoft Azure. The process for designing ML solution has been largely reduced to dragging and dropping customized components from the tool stack to the orchestration canvas and configuring a few steps in the solution workflow. It makes the process a lot easier as solutions can be easily churned out. These solutions are configured to have API endpoints, such that any deployed solution can be integrated with other systems in a real-time fashion.

It has capabilities to do data exploration, and can integrate with customized codes in Python, R-Language, C#, F#, Node.js and Java. Azure Machine Learning Studio was first released in 2015. A lot of improvement and capabilities have trickled in since the first release.

* + 1. **Python Interpreter**

One of the means of translating high level instructions into computer instructions is through an Interpreter. An Interpreter is a language translator that takes a whole high-level code and translates them to computer codes line by line. It does not translate the whole code all at once but does one instruction at a time.

Python codes are translated by a dedicated interpreter. Python language has become of the most popular modern programing language. It is widely used in mainstream programing for developing commercial apps, automating networking operations, scientific research and in artificial intelligence. Python is said to be easy to understand with simple syntaxes compared to other contemporary programing languages that follow a rigid and less intuitive syntaxes. Python programing language is reported to among the top three modern language. It has a large community of contributors who refine and modernize the language; they provide readily made and highly optimized libraries that programmers can use for several areas of discipline.

Python interpreter is free and can be downloaded from https: //www.python.org/downloads. The site also offers some opensource code editors that support python. Python interpreter has evolved from version 1.0 to its current version of 3.10.7. Each version adds new features, capabilities and address any reported vulnerabilities.

* + 1. **Code Editor (CE)**

Code editor is an integrated development environment that supports one or more programing technologies. CE may run an interpreter or compiler or sometimes both. CE are supposed to be language agnostic that is they can support multiple languages at the same time. Modern day code editors do have many advance features and capabilities like marketplace, elegant error handling, code and git management, refactoring, automated deployment, and self-service. For instance, marketplace allows the programmer to subscribe and download plugins on the code editor. Also, it shows updates on the plugins. Code and git management allows team of developers to collaborate and deliver values quickly. Also, it allows the team to enforce certain security standards and coding best practices. Deployment effort can be automated from an integrated code editor; usually they have interfaces with the deployment infrastructures.

Python Interpreter is supported by most modern code editors because of its popularity and applicability. Some code editors that support Python are **Visual studio code, Microsoft Visual studio, Atom, Sublime, NotePad++, Vim, Brackets, TextMate** etc. The choice of a code editor should be based on provision of features and capabilities that can bring about high productivity and better process management.

Most of these code editors require the installation of a python extension because each time a program is run, an extension must be assigned to run that instance of execution.

**Chapter Four**

**Implementation and Result**

* 1. **Implementation**

**This section explains how the methodology was followed to achieve the desired solution.**

* + 1. **Datasets**

**The project involves two datasets that are needed for the simulation: distance matrix and average flow rate matrix. The distance dataset captures the distance or weight between any two connecting points in the network while the average flow rate captures the average flow between any two points. Dash (-) indicates no connection between the two points that form the intersection. See table 6 and 7 for distance and average flow rate respectively.**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** |
| **A** | - | 10 | 4 | - | 8 | - | - | - | - | - | - |
| **B** | 10 | - | - | 5 | - | 8 | - | - | - | - | - |
| **C** | 4 | - | - | - | 5 | - | 12 | - | - | - | - |
| **D** | - | 5 | - | - | 4 | - | - | - | - | - | - |
| **E** | 8 | - | 5 | 4 | - | 5 | 5 | 6 | - | - | - |
| **F** | - | 8 | - | - | 5 | - | - | - | 3 | - | 13 |
| **G** | - | - | 12 | - | 5 | - | - | - | - | - | 13 |
| **H** | - | - | - | - | 6 | - | - | - | - | 6 | - |
| **I** | - | - | - | - | - | 3 | - | - | - | 7 | - |
| **J** | - | - | - | - | - | - | - | 6 | 7 | - | 3 |
| **K** | - | - | - | - | - | - | 13 | - | - | 3 | - |
|  |  |  |  |  |  |  |  |  |  |  |  |

Table 7: Distance Matrix

* + 1. **Road Network Design**

**This section explains how the road network was implemented using the python programing language. The distance matrix file dubbed “dsmx.txt” contains eleven (11) rows and columns which represent the intersections between two or more roads. Points that are connected to each other would have a value greater than zero if otherwise it will have a value zero. It is a comma delimited file was loaded by the program to generate a design that looks like figure 6 above.**

**Pyplot is a popular python library for plotting chats and graphical contents in Python.**

* + 1. **Coding**
    2. **Azure Machine Learning Integration**
  1. **Results and Discussions**

**Chapter Five**

**Conclusion and Recommendation**

* 1. **Conclusion**

**Djdjd**

* 1. **Recommendation and Future Work.**

**llll**