A Simulation Framework for a Real-Time Demand Responsive Public Transit System*

Thilina Perera, Chathura Nagoda Gamage, Alok Prakash and Thambipillai Srikanthan

Abstract— **Transit systems have encountered a radical change in the recent past as a result of the digital disruption. Consequently, traditional public transit systems no longer satisfy the diversified demands of passengers and hence, have been complemented by demand responsive transit solutions. However, we identify a lack of simulation tools developed to test and validate complex scenarios for real-time demand responsive public transit. Thus, in this paper, we propose a simulation framework, which combines complex scenario creation, optimization algorithm execution and result visualization using SUMO, an open source continuous simulator. In comparison to a state-of-the-art work, the proposed tool supports features such as varying vehicle capacity and driving range, immediate and advance passenger requests and maximum travel time constraints. Further, the framework follows a modular architecture that allows plug-and-play support for external modules.**

I. Introduction

Digital disruption has induced a wave of novel business models in the recent past. The effects of digital disruption is evident in all spheres of human life including public transit. As a result, traditional timetable-based transit systems no longer satisfy the diversified demands of passengers and hence, innovative real-time demand responsive transit (DRT) systems have become the future of public transit.

DRT systems have gained widespread popularity not only due to the ability to satisfy multiple user-groups but also by extending the penetration of existing public transit by linking the critical first/last miles. However, most of DRT-based work has been focused on ride-sourcing systems (e.g. Uber, Grab) which connect private hire drivers with passengers using a mobile application [1]. In contrast, relatively new research such as the work by Javier et al. [2] has lead to the emergence of real-time DRT-based public transit systems. Such systems typically consist of a fleet of buses, which operate with flexible routes and timetables. These services have recently been proposed in Australia [3] and Singapore [4].

Further, existing work on DRT-based public transit, focus mostly on developing optimal routing and scheduling algorithms [5] [6] [7]. However, prior to the implementation of algorithms using actual vehicles, it is essential that such a system can be tested and validated using simulations [8]. This not only saves a considerable amount of time but also reduces the cost of implementation. Furthermore, simulations can help to easily create realistic scenarios which can occur in real-world implementation. Also, evaluation and demonstration of a proposed idea prior to implementation and testing and training the users through simulation models can immensely benefit transit service providers [9]. Additionally, simulations help to visualize the flow of vehicles in order to understand their behavior. Also, the complexity of describing some systems using analytical models calls for simulation models. Thus, we identify the benefits of developing a simulation framework that can generate complex scenarios to test and validate DRT-based public transit systems.

However, as shown later in Section II, existing tools lack support to automatically generate complex scenarios to test DRT-based public transit systems. Hence, this work attempts to bridge the gap by proposing a simulation framework to test different scenarios using a real-world virtual environment. The main **contributions** of this work are (1) development of a holistic framework, which combines complex scenario creation, optimization algorithm execution and result visualization using an open source continuous simulator (2) a plug-and-play architecture that facilitates rapid turn-around time for replacement of the modules using external tools. The rest of the paper is organized as follows. Section II highlights the state-of-the-art work, followed by the architecture of the framework in Section III and an illustration of a simulation in Section IV. Finally, we conclude in Section V.

II. Related Work

Simulation models are classified as macroscopic, mesoscopic and microscopic models based on the level of details incorporated in the model [10]. A macroscopic model describes entities and their activities and interactions at a low level of detail. In contrast, a microscopic model describes both the system entities and their interactions at a high level of detail. Mesoscopic models generally represent entities at a high level of detail but describes their interactions at a much lower level of detail. However, in the case of DRT-based systems, each vehicle needs to be controlled independently and hence, microscopic simulations are preferred.

Existing microscopic traffic simulation packages such as PARAMICS [11], VGSim [12], SUMO [13] and Aimsun [14] provide functionality for route choice, traffic light control, vehicle communication, demand modeling, etc. Further, these tools allow users to simulate road networks of a given map. However, as a result of the open source licensing and extensive support and features, our simulation framework is built using SUMO (Simulation of Urban MObility), which has already been used in similar works [13].

*This research project is partially funded by the National Research Foundation Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) program with the Technical University of Munich at TUMCREATE.

Authors are with School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798. pere0004@e.ntu.edu.sg, chathuratng@gmail.com, alok@ntu.edu.sg, astsrikan@ntu.edu.sg

Macedo et al. [15] propose a model to simulate electric buses in a realistic urban mobility environment. They propose to couple SUMO with a model of an electric vehicle generated using MatLab/Simulink. However, the main goal of this work is to model electric vehicle behavior and hence, is not comparable with the proposed simulation framework. Similarly, Mayer et al [16] propose a simulation model and its corresponding discrete event simulator for a rich vehicle routing problem. Bischoff et al [17] propose an approach to dynamically simulate ridesharing services with a fleet of vehicles of a capacity between two and four. The work that is closest to our work is proposed in [8]. Here, authors propose a simulation platform developed using Vissim to test vehicle routing algorithms in a fleet management system. Further, it allows users to perform a graphical illustration of vehicles serving on-demand customer requests. However, the illustration is based on only a single vehicle and 5 customer requests. Also, it does not model features such as maximum travel time constraints, advanced passenger requests etc.

III. Simulation Framework

In this section, we first present the simulation setup followed by the simulation framework. The framework consists of three modules, namely the graphical user interface (GUI), vehicle routing and scheduling engine (VRSE) and the open source simulation platform (OSSP). The GUI facilitates to build a test scenario and subsequently the VRSE generates the routes and schedules of the vehicles. Finally, the OSSP simulates the test scenario. Subsequent sections explain the architecture of the modules in detail.

A. Simulation Setup

We select the locality surrounding Nanyang Technological University (NTU), Singapore as the virtual environment to create test scenarios. Within this locality, we assume passengers request for rides using a smartphone application from their respective origin locations. Then, a vehicle/vehicles from the fleet will be scheduled in real-time to pick-up the passengers and drop-off at the destination without violating the constraints of the system. However, it should be noted that, in the illustrations provided in the subsequent sections, we have limited the passenger request origins to existing bus stop locations (nodes). Also, for clarity, we have only shown 5 nodes. However, the methods proposed in this work are independent of the number of nodes. Henceforth, we will use this setup to illustrate the functionalities of the simulator.

B. Graphical User Interface

The GUI is used to create an instance of the problem by distributing passengers and vehicles on the map. Further, the GUI allows to configure microscopic parameters of both passengers and vehicles. Next, the generated scenario is validated and exported to the VRSE. After the VRSE execution, results are imported into the GUI for visualization. The flow diagram of the GUI is given in Fig. 1. Here, we have separated the GUI module into 2 submodules, namely *Scenario Creation* and *Data Export*. The former submodule

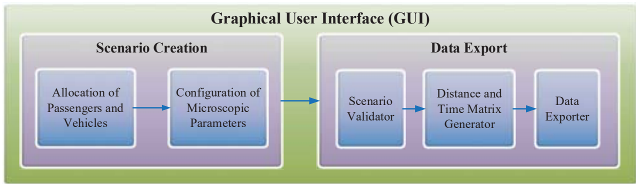

Graphical User Interface (GUI)
Scenario Creation
Data Export
Allocation of Passengers and Vehicles
Configuration of Microscopic Parameters
Scenario Validator
Distance and Time Matrix Generator
Data Exporter

Fig. 1: Flow Diagram of the GUI

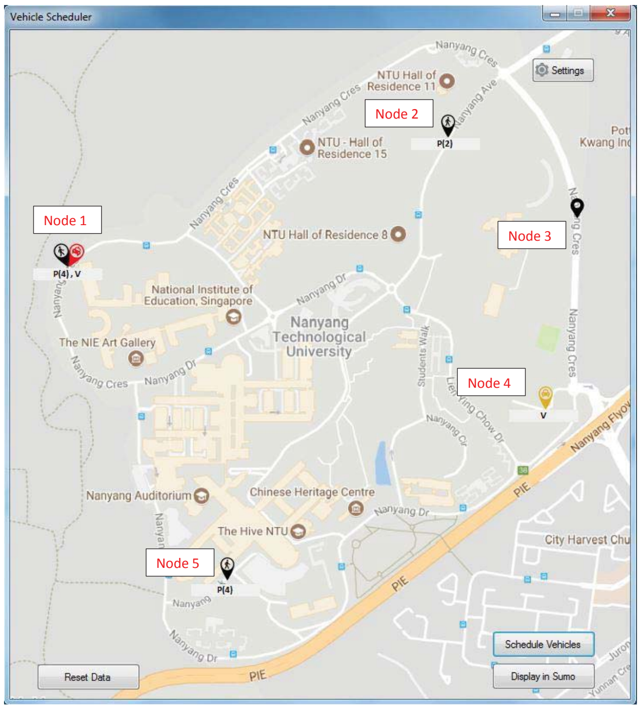


Fig. 2: Passenger and Vehicle Distribution

is managed by the user while the latter is handled by the tool. Also, it should be noted that we use Google Maps [18] to import the map of NTU to the GUI.

1) Scenario Creation: Scenario creation submodule consists of allocation and configuration of microscopic parameters of passengers and vehicles. Fig. 2 shows an allocation of 10 passengers and 2 vehicles in the GUI. Also, the corresponding description of terms used in the GUI is given in Table I. Subsequently, microscopic parameters can be configured using the *Settings* button in the GUI.

Microscopic parameters that can be configured are capacity of vehicles, driving range of vehicles (for simulation of electric vehicles with limited driving range), passenger request start time (immediate requests or advance requests) and maximum travel time of passengers (waiting time + riding time). Also, the parameters can be further configured to generate homogeneous or heterogeneous configurations. Thus, if a user sets a parameter to the homogeneous configuration mode, the corresponding value of that parameter will be equal in all objects. In contrast, if the user chooses the heterogeneous configuration mode, the corresponding value of that parameter will vary. Table II provides a description of the microscopic parameters for both configuration modes and the respective variables that can be configured by a user.

TABLE I: Description of Terms in the GUI

Term	Description	Example
$P(n)$	Node with n passengers	Node 2
V	Node with a vehicle	Node 4
$P(n), V$	Node with n passengers and a vehicle	Node 1
No Description	Node without passengers or a vehicle	Node 3

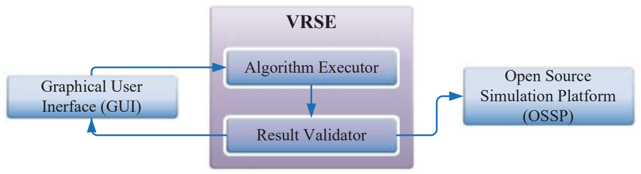

VRSE
Graphical User Inerface (GUI)
Algorithm Executor
Result Validator
Open Source Simulation Platform (OSSP)

Fig. 3: Flow Diagram of the VRSE

2) Data Export: Data export submodule consists of scenario validator, distance and time matrix generator and the data exporter. Validation of the scenario is performed as a prerequisite of the VRSE module. VRSE may have specific input conditions that are mandatory for accurate functioning. For example, if the optimization algorithm satisfies all passenger requests, the vehicles should have sufficient capacity. Hence, we validate this condition prior to VRSE execution. However, it is still feasible to execute VRSE even without the presence of the scenario validator.

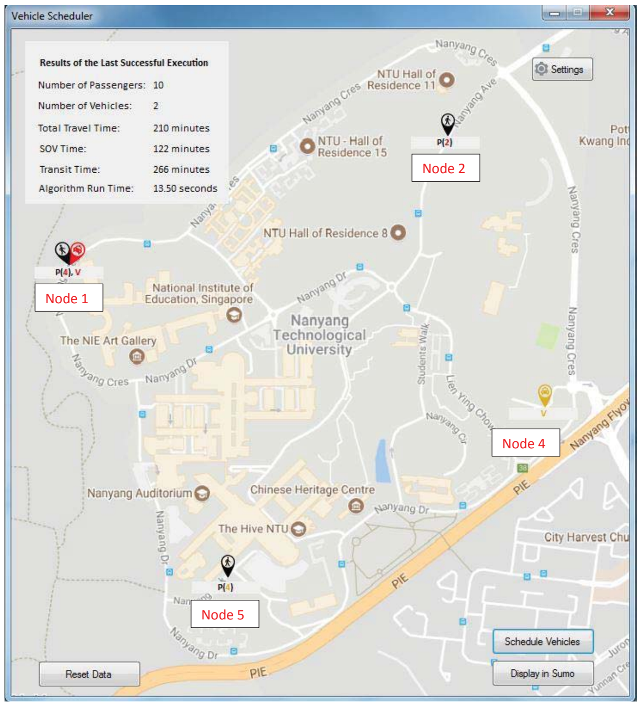
Next, the distance and time matrices are generated by querying the Google distance matrix API [19]. It should be noted that, this API generates travel times considering the real-time traffic conditions of the road network. Also, we extract the transit times (travel time to the destination using existing public transit) of the passengers. Finally, the data exporter converts the test scenario into the VRSE compliant file formats. It is noteworthy, that the modular architecture of the framework allows a user to plug-and-play a third party VRSE module. To this end, a user can either comply to the current output format of the data exporter or change the data exporter according to the third party VRSE file format.

C. Vehicle Routing and Scheduling Engine

Task of VRSE is generating the routes and schedules of the vehicles for the test scenario. The flow diagram of the VRSE module is given in Fig. 3. It comprises of two submodules, namely *Algorithm Executor* and *Result Validator*.

1) Algorithm Executor: The algorithm executor is the main computation engine of the simulator. It accepts the input data from the GUI and generates routes and schedules based on the optimization algorithm. In the illustrations, we use the optimization algorithm proposed in [20], which strives to minimize the travel time of passengers within the given constraints. However, as mentioned in Section III-B.2, a user can replace this module by a third party component.

2) Result Validator: This is an optional submodule which automates the validation of the routes and schedules generated by the algorithm executor. Based on the optimization algorithm in [20], we validate the following constraints, (i) all passengers are serviced by the vehicles, (ii) each passenger is serviced by only one vehicle, (iii) capacity constraint of the vehicles are not violated and finally (iv)


Vehicle Scheduler

Results of the Last Successful Execution

Number of Passengers:	10
Number of Vehicles:	2
Total Travel Time:	210 minutes
SOV Time:	122 minutes
Transit Time:	266 minutes
Algorithm Run Time:	13.50 seconds

Fig. 4: Results of the VRSE Execution

vehicle miles traveled by each vehicle is within driving range constraints. After validation of the results, they are exported to the GUI for visualization (Refer Section III-C.3) and the OSSP for simulation (Refer Section III-D).

3) Visualization of Results in the GUI: An illustration of the outcome of the VRSE for the test scenario in Section III-B.1 with 10 passengers and 2 vehicles is shown in Fig. 4. As mentioned in Section III-C.1, based on the objective of the optimization algorithm used in VRSE to minimize travel time, the output that is shown in the GUI compares temporal features. It also shows a summary of the test scenario and the runtime of the optimization algorithm. Table III provides a description of the results in the GUI.

In this example, the total travel time of the passengers using the proposed DRT solution is 210 minutes. In contrast, if passengers use private vehicles (SOV) to travel to the destination the travel time is 122 minutes. The SOV time is the lower bound of the outcome of the test scenario. Similarly, the transit time reflects an upper bound. The quality of optimization algorithm can be easily analyzed by comparing the result with these two values using the GUI.

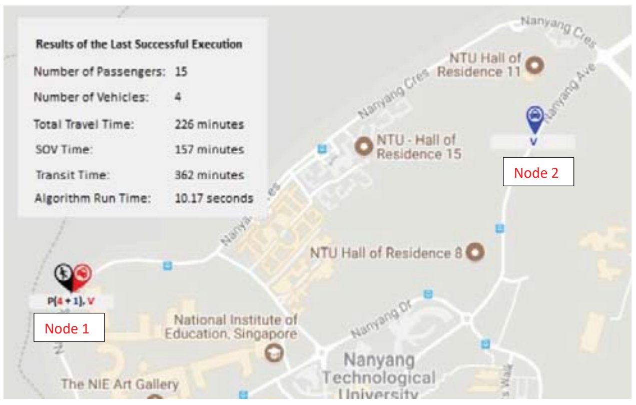
Another feature of the GUI is the color-based visualization scheme for displaying results. We illustrate this using Fig. 5, a segment of an output for a test scenario. Here, we observe two vehicles at nodes 1 and 2, displayed in red and blue color respectively. In this, four passengers at node 1 have been allocated to the vehicle shown in red while the other passenger is allocated to the vehicle shown in blue. This is represented in Fig. 5, as $P(4+1)$ with the corresponding colors of the two vehicles. Thus, the color-based visualization scheme helps quick identification of the outcome.

TABLE II: Configuration Modes of Microscopic Parameters

Microscopic Parameter	Homogeneous Configuration	Heterogeneous Configuration
Vehicle Capacity	Constant value of the vehicle capacity c : Capacity of all vehicles	Random value is assigned within the user-defined lower and upper bound values of the capacity C_1 : Lower bound of the capacity C_2 : Upper bound of the capacity
Vehicle Range	Constant value of the vehicle driving range r : Driving range of all vehicles	Sum of direct travel distance to the destination of all the passengers is averaged over the number of vehicles and randomly varied within $\pm$ R% R : Average travel distance variation percentage
Passenger Request Start Time	Constant value of the request start time of passengers 0 : Immediate request t : Advance request valid after the t^{th} minute	Random value is assigned within the user-defined lower and upper bound values of the request start time T_1 : Lower bound of the request start time T_2 : Upper bound of the request start time
Maximum Travel Time	Constant value of the maximum travel time of passengers n : Maximum travel time	Direct travel time to the destination of each passenger is incremented by N% N : Direct travel time increment percentage

TABLE III: Description of Results in the GUI

Term	Description
Total Travel Time	Total travel time of the passengers using the proposed DRT solution for the optimization algorithm used in VRSE module
SOV Time	Total travel time of the passengers using a private vehicle (Single Occupancy Vehicle) to travel to the destination
Transit Time	Total travel time of the passengers using public transit to travel to the destination
Algorithm Runtime	The runtime of the optimization algorithm



Results of the Last Successful Execution

Number of Passengers:	15
Number of Vehicles:	4
Total Travel Time:	226 minutes
SOV Time:	157 minutes
Transit Time:	362 minutes
Algorithm Run Time:	10.17 seconds

Fig. 5: Color-based Visualization in the GUI

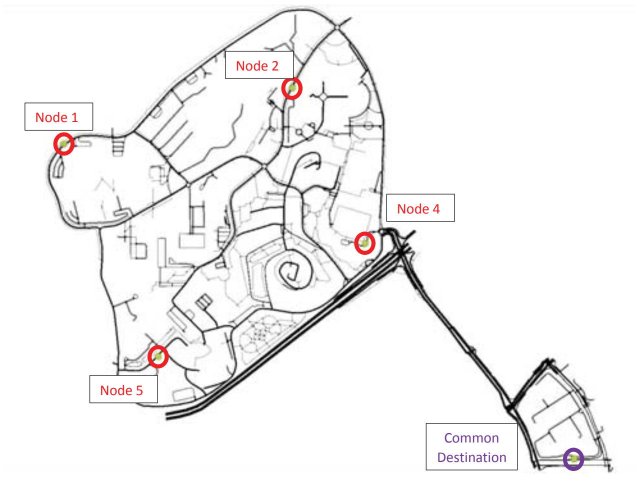
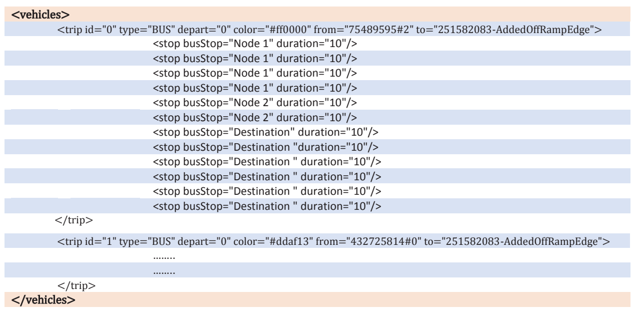


Fig. 6: Road Network used for Simulation using SUMO



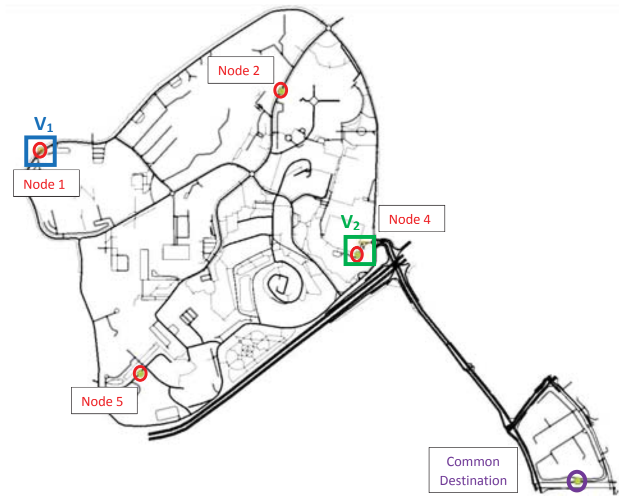
```
<vehicles>
        <trip id="0" type="BUS" depart="0" color="#ff0000" from="75489595#2" to="251582083-AddedOffRampEdge">
                        <stop busStop="Node 1" duration="10"/>
                        <stop busStop="Node 1" duration="10"/>
                        <stop busStop="Node 1" duration="10"/>
                        <stop busStop="Node 1" duration="10"/>
                        <stop busStop="Node 2" duration="10"/>
                        <stop busStop="Node 2" duration="10"/>
                        <stop busStop="Destination" duration="10"/>
                        <stop busStop="Destination "duration="10"/>
                        <stop busStop="Destination " duration="10"/>
                        <stop busStop="Destination " duration="10"/>
                        <stop busStop="Destination " duration="10"/>
                        <stop busStop="Destination " duration="10"/>
        </trip>
        <trip id="1" type="BUS" depart="0" color="#ddaf13" from="432725814#0" to="251582083-AddedOffRampEdge">
                        ........
                        ........
        </trip>
</vehicles>
```

Fig. 7: Sample Routes of the Trip Generator

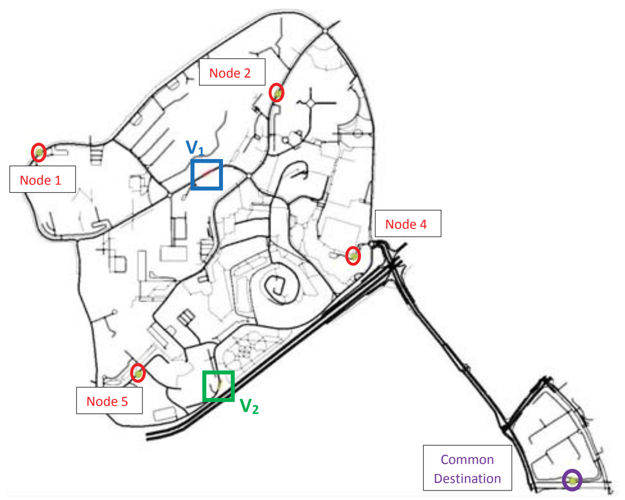
D. *Open Source Simulation Platform*

The OSSP module performs a continuous simulation of the routes generated by the VRSE. The OSSP comprises 5 submodules, namely *Road Network Generator*, *Fleet and Node Manager*, *Trip Generator*, *Visual Configuration Manager* and *Routing Manager*. Here, the *Road Network Generator* creates the road network of the selected area. Properties of the fleet of vehicles and pick-up points are configured in the *Fleet and Node Manager*. The *Trip Generator* submodule generates the routes of the vehicles for the simulation based on VRSE results. Next, the *Visual Configuration Manager* sets the graphical settings in the SUMO GUI. Finally, the *Routing Manager* performs the continuous simulation according to the microscopic parameters configured by the user. The submodules are explained in detail in the subsequent sections.

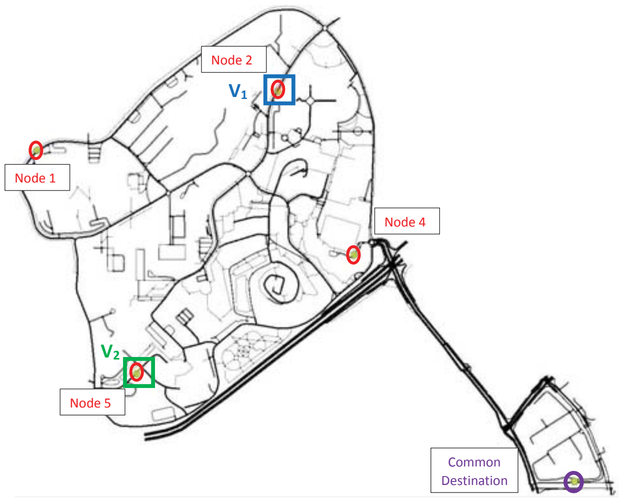
1) Road Network Generator: This submodule generates the road network for the setup given in Section III-A. Thus, for the simulation, we generate the road network for the map of NTU shown in Fig. 2. Here, we use Open Street Maps (OSM) [21] to obtain the map data of the selected area. The quality of the simulation can be enhanced further by processing the map data obtained from OSM using the Java OpenStreet Map editor (JOSM) [22]. Therefore, we have improved the connectivity of the road network using JOSM. Further, a user can configure the speed limits of the roads, the number of lanes, one-way roads and traffic light systems using the JOSM editor. Next, we convert the map into a SUMO road network [23] file using NETCONVERT [24]. During this process, the map is converted to a directed graph such that traffic can be simulated on it using SUMO. Further, NETCONVERT provides options to discard certain types of roads, remove roads that are not connected etc. Finally, we



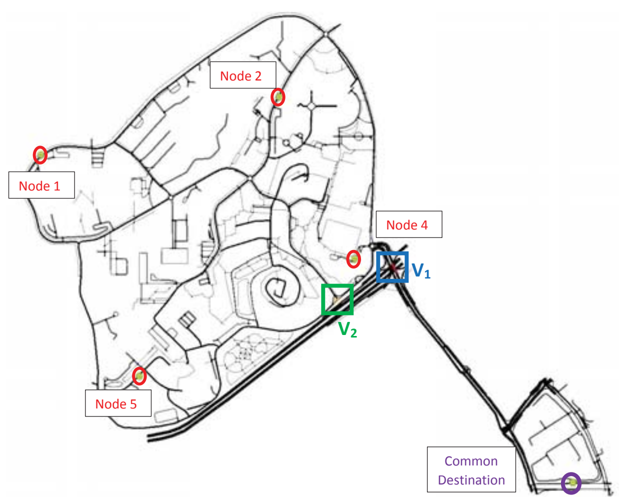
(a) Origins; V_1: Node 1, V_2: Node 4



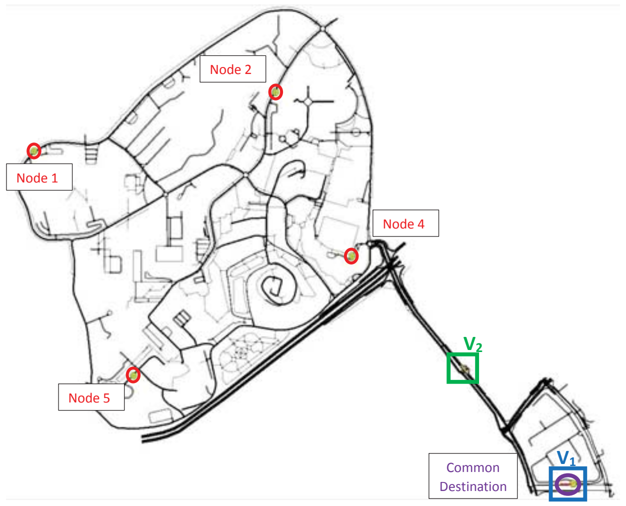
(b) Route to Pick-up Points; V_1: Node 1 $\longrightarrow$ Node 2, V_2: Node 4 $\longrightarrow$ Node 5



(c) At Pick-up Points; V_1: Node 2, V_2: Node 5



(d) Route to Destination; V_1: Node 2 $\longrightarrow$ Common Destination, V_2: Node 5 $\longrightarrow$ Common Destination



(e) At Destination; V_1: Common Destination V_2: $\longrightarrow$ Common Destination

Fig. 8: Time Interleaved Continuous Simulation Results

generate a map suitable for traffic simulation using SUMO.

2) Fleet and Node Manager: This submodule defines the properties of the vehicles used in the simulation. It configures vehicle attributes such as acceleration, deceleration, maximum speed etc. Also, it is used to include the passenger pick-up and drop-off points to the generated map. The exact locations of the pick-up points are extracted from the locations selected by the user when creating the instance in Fig. 4. However, according to the optimization algorithm in [20], the drop-off point of all passengers is a common destination. The road network generated for the map of NTU in Fig. 2 after pre-processing is given in Fig. 6. Here, we have highlighted the passenger pick-up locations in red circles and the common destination with a purple circle. It should be noted that node 3 is not shown in this network as it is not a pick-up point.

3) Trip Generator: After creating the road network and adding the vehicles and passengers to the map, the next step is assigning the routes for individual vehicles. Each route starts from the origin of the vehicles and ends at the common destination. Along the route, vehicles will visit the passengers to pick them up. Thus, output of the VRSE is used to create the route file. We generate an XML file which includes the routes for each vehicle. Fig. 7 shows a sample route file generated for the results in Fig. 4. The route file contains two entries for a passenger, one for the pick-up point and the other for the drop-off point. Hence, the route file for V_1 consists of 12 entries, corresponding to 6 passengers. Further, at each pick-up and drop-off point we add a constant (10 time unit) waiting time. This is reflected in all entries of the file.

4) Visual Configuration Manager: This submodule configures the graphical settings of the simulation. Here, visual attributes such as colors of the streets, vehicles, pick-up points and their names, display sizes etc. can be configured.

5) Routing Manager: Finally, the continuous simulation of the test scenario is managed by the SUMO routing engine. Here, we use Dijkstra algorithm to generate the shortest path between the nodes for routing the vehicles. This is in line with the distance matrix API used in VRSE to generate distance and time matrices, that produce the shortest paths.

IV. Simulation Results

This section shows an illustration of a simulation. The user can perform a simulation by clicking the *Display in SUMO* button in the GUI. At this point, the result generated from VRSE is pre-processed using the OSSP to perform a graphical simulation using SUMO. Here, we use the example in Fig. 4 for the illustration. The routes of the two vehicles as generated by the VRSE are V_1 : Node 1 $\longrightarrow$ Node 2 $\longrightarrow$ Common Destination; and V_2: Node 4 $\longrightarrow$ Node 5 $\longrightarrow$ Common Destination. Fig. 8 shows a time interleaved simulation of the vehicle routes. Here, we have highlighted the two vehicles V_1 and V_2 using blue and green squares respectively. Also, the nodes and the common destination are also marked. Fig. 8a shows the two vehicles at the origins of the routes. Fig. 8b shows the two vehicles moving to the passenger pick-up points, namely node 2 and node 5 respectively. Fig. 8c shows both vehicles at the respective pick-up points. Next, Fig. 8d shows both vehicles moving to the common destination. Finally, Fig. 8e shows that V_1 has already reached the common destination while V_2 is still moving along the route to the common destination.

V. Conclusion

In this work, we propose a simulation framework, which combines complex scenario creation, optimization algorithm execution and result visualization using the simulator SUMO. The framework helps to test, validate and visualize real-time DRT-based public transit systems, which is a hot topic of research. The framework has a modular architecture and hence support plug-and-play capability for external modules. In future, we plan to incorporate traffic information, pedestrian behavior etc. in order to illustrate a more realistic simulation.

References

[1] R. Clavel *et al.*, "The role of intelligent transport systems for demand responsive transport," 2012.

[2] J. Alonso-Mora *et al.*, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *PNAS*, 2017.

[3] Transit Systems: Demand Responsive Services, "Demand Responsive bus services to be launched," https://www.transitsystems.com.au/demand-responsive-transport-systems/, 2017.

[4] Land Transport Authority, Singapore, "LTA awards first phase of tender for on-demand bus service trial," https://www.lta.gov.sg/apps/news/page.aspx?c=2&id=262fd468-2633-45ee-94ab-0274aae856d0, 2018.

[5] L. M. Martínez *et al.*, "Formulating a new express minibus service design problem as a clustering problem," *Transportation Science*, 2015.

[6] N. Marković *et al.*, "Optimizing dial-a-ride services in maryland: Benefits of computerized routing and scheduling," *Transportation Research Part C: Emerging Technologies*, 2015.

[7] C. Archetti *et al.*, "A simulation study of an on-demand transportation system," *International Transactions in Operational Research*, 2018.

[8] S. C. Nagavarapu *et al.*, "Development of a simulation platform to implement vehicle routing algorithms for large scale fleet management systems," in *ITSC*, 2017.

[9] M. Pursula, "Simulation of traffic systems - an overview," *Journal of Geographic Information and Decision Analysis*, 1999.

[10] T. V. Mathew, "Microscopic Traffic Simulation," https://www.civil.iitb.ac.in/tvm/nptel/tselnw37.pdf, 2017.

[11] G. D. B. Cameron and G. I. D. Duncan, "Paramics—parallel microscopic simulation of road traffic," *The Journal of Supercomputing*, 1996.

[12] B. Liu *et al.*, "Vgsim: An integrated networking and microscopic vehicular mobility simulation platform," *IEEE Comm. Magazine*, 2009.

[13] D. Krajzewicz *et al.*, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, 2012.

[14] Aimsun, "Move Brilliantly," https://www.aimsun.com/, 2018.

[15] J. Macedo *et al.*, "A hla-based multi-resolution approach to simulating electric vehicles in simulink and sumo," in *ITSC*, 2013.

[16] T. Mayer *et al.*, "An open-source discrete event simulator for rich vehicle routing problems," in *ITSC*, 2016.

[17] J. Bischoff *et al.*, "City-wide shared taxis: A simulation study in berlin," in *ITSC*, 2017.

[18] Google, "Google Maps," https://www.google.com/maps, 2018.

[19] "Google Maps APIs: Build the next generation of location experiences," https://developers.google.com/maps/.

[20] T. Perera *et al.*, "Hybrid genetic algorithm for an on-demand first mile transit system using electric vehicles," in *ICCS*, 2018.

[21] "Open Street Maps," https://www.openstreetmap.org, 2018.

[22] "Java Open Street Map Editor," https://josm.openstreetmap.de/, 2018.

[23] "Networks/SUMO Road Networks," http://sumo.dlr.de/wiki/Networks/SUMO_Road_Networks, 2018.

[24] "NETCONVERT," http://sumo.dlr.de/wiki/NETCONVERT, 2018.