PROJECT 3 REPORT

GROUP 2

OBJECTIVE

The goal of the AMoD challenges is to create an operational policy that maximizes the service level of the AMoD system while minimizing its operational cost. A well-performing autonomous mobility-on-demand system uses as few taxis as possible to pickup and deliver as many customers as possible with minimal waiting and journey times while keeping the fleet mileage to a minimum.

ALGORITHM BEING USED

In an autonomous mobility-on-demand system a coordinated fleet of robotic taxis serves customers in an on-demand fashion. An operational policy for the system must optimize in three conflicting dimensions:

- The system must perform at the highest possible service level, i.e., at smallest possible wait times and smallest possible journey times.
- The system's operation must be as efficient as possible, i.e., it must reduce its empty mileage to a minimum.
- The system's capital cost must be as inexpensive as possible, i.e., the fleet size must be reduced to a minimum.

We consider robotic taxis that can carry one customer. To compare different AMoD system operational policies, we introduce the following variables:

$$
\begin{aligned}
&d_E && = \text{empty distance driven by the fleet} \\
&d_C && = \text{occupied distance driven by the fleet} \\
&d_T = d_C + d_E && = \text{total distance driven by the fleet} \\
&N && = \text{fleet size} \\
&R && = \text{number of customer requests served} \\
&w_i && = \text{waiting time of request } i \in \{1, \ldots, R\} \\
&W && = \text{total waiting time } W = \sum_{i=1}^{R} w_i
\end{aligned}
$$

The principal goal is to perform as efficiently as possible while maintaining the best possible service level. Two negative scalar weights α3<0 and α4<0 are introduced. The performance metric to maximize is

$$\mathcal{J}_{P-AMoD,2} = \alpha_3 W + \alpha_4 d_E$$

α3 and α4 are chosen such that the term d_E dominates the metric. The number of robotic taxis is fixed at some fleet size ¯N∈N>0.

CODE

SCENARIO PREPARER

```
package amod.demo;
import java.io.File;
import java.net.MalformedURLException;

import org.matsim.api.core.v01.Scenario;
import org.matsim.api.core.v01.network.Network;
import org.matsim.api.core.v01.population.Population;
import org.matsim.core.config.Config;
import org.matsim.core.config.ConfigUtils;
import org.matsim.core.scenario.ScenarioUtils;

import amod.demo.ext.Static;
import ch.ethz.idsc.amodeus.options.ScenarioOptions;
import ch.ethz.idsc.amodeus.options.ScenarioOptionsBase;
import ch.ethz.idsc.amodeus.prep.ConfigCreator;
import ch.ethz.idsc.amodeus.prep.NetworkPreparer;
import ch.ethz.idsc.amodeus.prep.PopulationPreparer;
import ch.ethz.idsc.amodeus.prep.VirtualNetworkPreparer;
import ch.ethz.idsc.amodeus.util.io.MultiFileTools;
import ch.ethz.idsc.amodeus.util.io.ProvideAVConfig;
import ch.ethz.matsim.av.config.AVConfig;
import ch.ethz.matsim.av.config.AVGeneratorConfig;
import ch.ethz.matsim.av.framework.AVConfigGroup;

/** Class to prepare a given scenario for MATSim, includes preparation
of
 * network, population, creation of virtualNetwork and travelData
objects. As an example
 * a user may want to restrict the population size to few 100s of
agents to run simulations
 * quickly during testing, or the network should be reduced to a
certain area. */
public enum ScenarioPreparer {
    ;

    public static void main(String[] args) throws
MalformedURLException, Exception {
        File workingDirectory =
MultiFileTools.getDefaultWorkingDirectory();
        run(workingDirectory);
    }

    /** loads scenario preparer in the {@link File} workingDirectory
@param workingDirectory
     *
     * @throws MalformedURLException
     * @throws Exception */
```

```
    public static void run(File workingDirectory) throws
MalformedURLException, Exception {
        Static.setup();
        Static.checkGLPKLib();

        /** The {@link ScenarioOptions} contain amodeus specific
options. Currently there are 3
         * options files:
         * - MATSim configurations (config.xml)
         * - AV package configurations (av.xml)
         * - AMoDeus configurations (AmodeusOptions.properties).
         *
         * The number of configs is planned to be reduced in
subsequent refactoring steps. */
        ScenarioOptions scenarioOptions = new
ScenarioOptions(workingDirectory, ScenarioOptionsBase.getDefault());

        /** MATSim config */
        AVConfigGroup avConfigGroup = new AVConfigGroup();
        Config config =
ConfigUtils.loadConfig(scenarioOptions.getPreparerConfigName(),
avConfigGroup);
        Scenario scenario = ScenarioUtils.loadScenario(config);
        AVConfig avConfig = ProvideAVConfig.with(config,
avConfigGroup);
        AVGeneratorConfig genConfig =
avConfig.getOperatorConfigs().iterator().next().getGeneratorConfig();
        int numRt = (int) genConfig.getNumberOfVehicles();
        System.out.println("NumberOfVehicles=" + numRt);

        /** adaption of MATSim network, e.g., radius cutting */
        Network network = scenario.getNetwork();
        network = NetworkPreparer.run(network, scenarioOptions);

        /** adaption of MATSim population, e.g., radius cutting */
        Population population = scenario.getPopulation();
        long apoSeed = 1234;
        PopulationPreparer.run(network, population, scenarioOptions,
config, apoSeed);

        /** creating a virtual network, e.g., for operational policies
requiring a graph structure on the city */
        int endTime = (int) config.qsim().getEndTime();
        VirtualNetworkPreparer.INSTANCE.create(network, population,
scenarioOptions, numRt, endTime); //

        /** create a simulation MATSim config file linking the created
input data */
        ConfigCreator.createSimulationConfigFile(config,
scenarioOptions);
    }
}
```

SCENARIOSERVER

```
package amod.demo;

import java.io.File;
import java.net.MalformedURLException;
import java.util.Objects;

import org.matsim.api.core.v01.Scenario;
import org.matsim.api.core.v01.network.Network;
import org.matsim.api.core.v01.population.Population;
import org.matsim.contrib.dvrp.run.DvrpConfigGroup;
import org.matsim.contrib.dvrp.trafficmonitoring.DvrpTravelTimeModule;
import org.matsim.core.config.Config;
import org.matsim.core.config.ConfigUtils;
import
org.matsim.core.config.groups.PlanCalcScoreConfigGroup.ActivityParams;
import org.matsim.core.controler.AbstractModule;
import org.matsim.core.controler.Controler;
import org.matsim.core.scenario.ScenarioUtils;

import com.google.inject.Key;
import com.google.inject.name.Names;

import amod.demo.analysis.CustomAnalysis;
import amod.demo.dispatcher.DemoDispatcher;
import amod.demo.ext.Static;
import amod.demo.generator.DemoGenerator;
import ch.ethz.idsc.amodeus.analysis.Analysis;
import ch.ethz.idsc.amodeus.data.LocationSpec;
import ch.ethz.idsc.amodeus.data.ReferenceFrame;
import ch.ethz.idsc.amodeus.linkspeed.LinkSpeedDataContainer;
import ch.ethz.idsc.amodeus.linkspeed.LinkSpeedUtils;
import ch.ethz.idsc.amodeus.linkspeed.TrafficDataModule;
import ch.ethz.idsc.amodeus.matsim.mod.AmodeusDatabaseModule;
import ch.ethz.idsc.amodeus.matsim.mod.AmodeusDispatcherModule;
import ch.ethz.idsc.amodeus.matsim.mod.AmodeusModule;
import ch.ethz.idsc.amodeus.matsim.mod.AmodeusVehicleGeneratorModule;
import
ch.ethz.idsc.amodeus.matsim.mod.AmodeusVehicleToVSGeneratorModule;
import ch.ethz.idsc.amodeus.matsim.mod.AmodeusVirtualNetworkModule;
import ch.ethz.idsc.amodeus.net.DatabaseModule;
import ch.ethz.idsc.amodeus.net.MatsimAmodeusDatabase;
import ch.ethz.idsc.amodeus.net.SimulationServer;
import ch.ethz.idsc.amodeus.options.ScenarioOptions;
import ch.ethz.idsc.amodeus.options.ScenarioOptionsBase;
import ch.ethz.idsc.amodeus.routing.DefaultAStarLMRouter;
import ch.ethz.idsc.amodeus.util.io.MultiFileTools;
import ch.ethz.idsc.amodeus.util.math.GlobalAssert;
import ch.ethz.matsim.av.framework.AVConfigGroup;
import ch.ethz.matsim.av.framework.AVModule;
```

```
import ch.ethz.matsim.av.framework.AVUtils;

/** This class runs an AMoDeus simulation based on MATSim. The results
can be viewed
 * if the {@link ScenarioViewer} is executed in the same working
directory and the button "Connect"
 * is pressed. */
public enum ScenarioServer {
    ;

    public static void main(String[] args) throws
MalformedURLException, Exception {
        simulate(MultiFileTools.getDefaultWorkingDirectory());
    }

    /** runs a simulation run using input data from
Amodeus.properties, av.xml and MATSim config.xml
     *
     * @throws MalformedURLException
     * @throws Exception */
    public static void simulate(File workingDirectory) throws
MalformedURLException, Exception {
        Static.setup();
        Static.checkGLPKLib();

        /** working directory and options */
        ScenarioOptions scenarioOptions = new
ScenarioOptions(workingDirectory, ScenarioOptionsBase.getDefault());

        /** set to true in order to make server wait for at least 1
client, for
         * instance viewer client, for fals the ScenarioServer starts
the simulation
         * immediately */
        boolean waitForClients =
scenarioOptions.getBoolean("waitForClients");
        File configFile = new
File(scenarioOptions.getSimulationConfigName());

        /** geographic information */
        LocationSpec locationSpec = scenarioOptions.getLocationSpec();
        ReferenceFrame referenceFrame = locationSpec.referenceFrame();

        /** open server port for clients to connect to */
        SimulationServer.INSTANCE.startAcceptingNonBlocking();
        SimulationServer.INSTANCE.setWaitForClients(waitForClients);

        /** load MATSim configs - including av.xml configurations,
load routing packages */
        GlobalAssert.that(configFile.exists());
        DvrpConfigGroup dvrpConfigGroup = new DvrpConfigGroup();
        dvrpConfigGroup.setTravelTimeEstimationAlpha(0.05);
```

```
        Config config = ConfigUtils.loadConfig(configFile.toString(),
new AVConfigGroup(), dvrpConfigGroup);
        config.planCalcScore().addActivityParams(new
ActivityParams("activity"));
        /** MATSim does not allow the typical duration not to be set,
therefore for scenarios
         * generated from taxi data such as the "SanFrancisco"
scenario, it is set to 1 hour. */
        for (ActivityParams activityParams :
config.planCalcScore().getActivityParams()) {
            // TODO set typical duration in scenario generation and
remove
            activityParams.setTypicalDuration(3600.0);
        }

        /** output directory for saving results */
        String outputdirectory =
config.controler().getOutputDirectory();

        /** load MATSim scenario for simulation */
        Scenario scenario = ScenarioUtils.loadScenario(config);
        Network network = scenario.getNetwork();
        Population population = scenario.getPopulation();
        GlobalAssert.that(Objects.nonNull(network));
        GlobalAssert.that(Objects.nonNull(population));

        MatsimAmodeusDatabase db =
MatsimAmodeusDatabase.initialize(network, referenceFrame);
        Controler controler = new Controler(scenario);
        controler.addOverridingModule(new DvrpTravelTimeModule());

        try {
            // load linkSpeedData if possible
            File linkSpeedDataFile = new
File(scenarioOptions.getLinkSpeedDataName());
            System.out.println(linkSpeedDataFile.toString());
            LinkSpeedDataContainer lsData =
LinkSpeedUtils.loadLinkSpeedData(linkSpeedDataFile);
            controler.addOverridingModule(new
TrafficDataModule(lsData));
        } catch (Exception exception) {
            System.err.println("Could not load static linkspeed data,
running with freespeeds.");
        }
        controler.addOverridingModule(new AVModule());
        controler.addOverridingModule(new DatabaseModule());
        controler.addOverridingModule(new
AmodeusVehicleGeneratorModule());
        controler.addOverridingModule(new AmodeusDispatcherModule());
        controler.addOverridingModule(new AmodeusDatabaseModule(db));
        controler.addOverridingModule(new
AmodeusVirtualNetworkModule(scenarioOptions));
```

```
        controler.addOverridingModule(new
AmodeusVehicleToVSGeneratorModule());
        controler.addOverridingModule(new AmodeusModule());
        controler.addOverridingModule(new AbstractModule() {
            @Override
            public void install() {
                bind(Key.get(Network.class,
Names.named("dvrp_routing"))).to(Network.class);
            }
        });

        /** With the subsequent lines an additional user-defined
dispatcher is added, functionality
         * in class
         * DemoDispatcher, as long as the dispatcher was not selected
in the file av.xml, it is not
         * used in the simulation. */
        controler.addOverridingModule(new AbstractModule() {
            @Override
            public void install() {
                AVUtils.registerDispatcherFactory(binder(), //
                        DemoDispatcher.class.getSimpleName(),
DemoDispatcher.Factory.class);
            }
        });

        /** With the subsequent lines, additional user-defined initial
placement logic called
         * generator is added,
         * functionality in class DemoGenerator. As long as the
generator is not selected in the
         * file av.xml,
         * it is not used in the simulation. */
        controler.addOverridingModule(new AbstractModule() {
            @Override
            public void install() {
                AVUtils.registerGeneratorFactory(binder(),
"DemoGenerator", DemoGenerator.Factory.class);
            }
        });

        /** With the subsequent lines, another custom router is added
apart from the
         * {@link DefaultAVRouter},
         * it has to be selected in the av.xml file with the lines as
follows:
         * <operator id="op1">
         * <param name="routerName" value="DefaultAStarLMRouter" />
         * <generator strategy="PopulationDensity">
         * ...
         *
```

```
         * otherwise the normal {@link DefaultAVRouter} will be used.
*/
        controler.addOverridingModule(new AbstractModule() {
            @Override
            public void install() {
                bind(DefaultAStarLMRouter.Factory.class);
                AVUtils.bindRouterFactory(binder(),
DefaultAStarLMRouter.class.getSimpleName())//
                        .to(DefaultAStarLMRouter.Factory.class);
            }
        });

        /** run simulation */
        controler.run();

        /** close port for visualizaiton */
        SimulationServer.INSTANCE.stopAccepting();

        /** perform analysis of simulation, a demo of how to add
custom
         * analysis methods is provided in the package
amod.demo.analysis */
        Analysis analysis = Analysis.setup(scenarioOptions, new
File(outputdirectory), network, db);
        CustomAnalysis.addTo(analysis);
        analysis.run();

    }
}
```

SCENARIOVIEWER

```
package amod.demo;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;

import org.matsim.api.core.v01.network.Network;
import org.matsim.core.config.Config;
import org.matsim.core.config.ConfigUtils;

import amod.demo.ext.Static;
import ch.ethz.idsc.amodeus.data.LocationSpec;
import ch.ethz.idsc.amodeus.data.ReferenceFrame;
import ch.ethz.idsc.amodeus.gfx.AmodeusComponent;
import ch.ethz.idsc.amodeus.gfx.AmodeusViewerFrame;
import ch.ethz.idsc.amodeus.gfx.ViewerConfig;
import ch.ethz.idsc.amodeus.matsim.NetworkLoader;
import ch.ethz.idsc.amodeus.net.MatsimAmodeusDatabase;
import ch.ethz.idsc.amodeus.options.ScenarioOptions;
import ch.ethz.idsc.amodeus.options.ScenarioOptionsBase;
import ch.ethz.idsc.amodeus.util.io.MultiFileTools;
import ch.ethz.idsc.amodeus.util.math.GlobalAssert;
import ch.ethz.idsc.amodeus.virtualnetwork.core.VirtualNetworkGet;

/** the viewer allows to connect to the scenario server or to view
saved simulation results. */
public enum ScenarioViewer {
    ;

    public static void main(String[] args) throws
FileNotFoundException, IOException {
        File workingDirectory =
MultiFileTools.getDefaultWorkingDirectory();
        run(workingDirectory);
    }

    /** Execute in simulation folder to view past results or connect
to simulation server
     *
     * @param args not used
     * @throws FileNotFoundException
     * @throws IOException */
    public static void run(File workingDirectory) throws
FileNotFoundException, IOException {
        Static.setup();
        ScenarioOptions scenarioOptions = new
ScenarioOptions(workingDirectory, ScenarioOptionsBase.getDefault());

        /** load options */
```

```
        Config config =
ConfigUtils.loadConfig(scenarioOptions.getSimulationConfigName());
        System.out.println("MATSim config file: " +
scenarioOptions.getSimulationConfigName());
        final File outputSubDirectory = new
File(config.controler().getOutputDirectory()).getAbsoluteFile();
        if (!outputSubDirectory.isDirectory()) {
            System.err.println("output directory: " +
outputSubDirectory.getAbsolutePath() + " not found.");
            GlobalAssert.that(false);
        }
        System.out.println("outputSubDirectory=" +
outputSubDirectory.getAbsolutePath());
        File outputDirectory = outputSubDirectory.getParentFile();
        System.out.println("showing simulation results from
outputDirectory=" + outputDirectory);

        /** geographic information, .e.g., coordinate system */
        LocationSpec locationSpec = scenarioOptions.getLocationSpec();
        ReferenceFrame referenceFrame = locationSpec.referenceFrame();

        /** MATSim simulation network */
        Network network = NetworkLoader.fromConfigFile(new
File(workingDirectory, scenarioOptions.getString("simuConfig")));
        System.out.println("INFO network loaded");
        System.out.println("INFO total links " +
network.getLinks().size());
        System.out.println("INFO total nodes " +
network.getNodes().size());

        /** initializing the viewer */
        MatsimAmodeusDatabase db =
MatsimAmodeusDatabase.initialize(network, referenceFrame);
        AmodeusComponent amodeusComponent =
AmodeusComponent.createDefault(db, workingDirectory);

        /** virtual network layer, should not cause problems if layer
does not exist */

amodeusComponent.virtualNetworkLayer.setVirtualNetwork(VirtualNetworkG
et.readDefault(network, scenarioOptions));

        /** starting the viewer */
        ViewerConfig viewerConfig = ViewerConfig.from(db,
workingDirectory);
        System.out.println("Used viewer config: " + viewerConfig);
        AmodeusViewerFrame amodeusViewerFrame = new
AmodeusViewerFrame(amodeusComponent, outputDirectory, network,
scenarioOptions);

amodeusViewerFrame.setDisplayPosition(viewerConfig.settings.coord,
viewerConfig.settings.zoom);
```

```
amodeusViewerFrame.jFrame.setSize(viewerConfig.settings.dimensions);
        amodeusViewerFrame.jFrame.setVisible(true);
    }
}
```

IMPORTANT PARAMETERS

AMODEUS PROPERTIES

completeGraph=true

dtTravelData=3600

FacilitiesUpdateName=preparedFacilities

fullConfig=config_full.xml

linkSpeedDataFileName=linkSpeedData

LocationSpec=SANFRANCISCO

maxPopulationSize=1000000

NetworkUpdateName=preparedNetwork

numVirtualNodes=10

populationchangeModeToAV=true

populationCutter=NETWORKBASED

PopulationUpdateName=preparedPopulation

shapeFile=AbsoluteShapeFileName

simuConfig=config.xml

travelDataFileName=travelData

virtualNetwork=virtualNetwork

virtualNetworkCreator=KMEANS

waitForClients=false

AV CONFIG

<av>

<param value="**-12.86**" name="**marginalUtilityOfWaitingTime**"/>

<timing>

<param value="**15.0**" name="**pickupDurationPerStop**"/>

<param value="**0.0**" name="**pickupDurationPerPassenger**"/>

<param value="**10.0**" name="**dropoffDurationPerStop**"/>

<param value="**0.0**" name="**dropoffDurationPerPassenger**"/>

</timing>

<operator id="**op1**">

<generator strategy="**PopulationDensity**">

<param value="**500**" name="**numberOfVehicles**"/>

</generator>

<dispatcher strategy="**DemandSupplyBalancingDispatcher**">

<param value="**10**" name="**dispatchPeriod**"/>

</dispatcher>

<pricing>

<param value="**0.001**" name="**pricePerKm**"/>

<param value="**0.0**" name="**pricePerMin**"/>

<param value="**3.0**" name="**pricePerTrip**"/>

<param value="**0.0**" name="**dailySubscriptionFee**"/>

</pricing>

</operator>

</av>

SIMULATION 1

NO OF VEHICLES: 450

DISPATCH PERIOD: 10

User:	dwang
Timestamp:	2019/05/02 - 00:47:10
Iterations:	1 ,i.e., 0 in matsim config.

AV File AV_Config File

Dispatcher:	DemandSupplyBalancingDispatcher
Rebalancing Period:	-00:00:01
Redispatching Period:	00:00:10
Network:	null_prepared
Virtual Nodes:	10 virtual nodes.
Population:	21562
Number of Vehicles:	450
Number of Requests	21562

Aggregated Results

Distance Ratio:	67.17%
Occupancy Ratio:	30.38 %

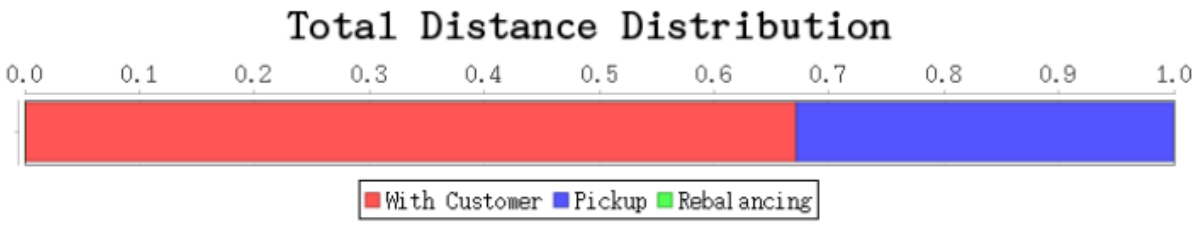
Distances

Total:	458536.01 km
Rebalancing:	0.00 km (0.00%)
Pickup:	150534.52 km (32.83%)
With Customer:	308001.49 km (67.17%)

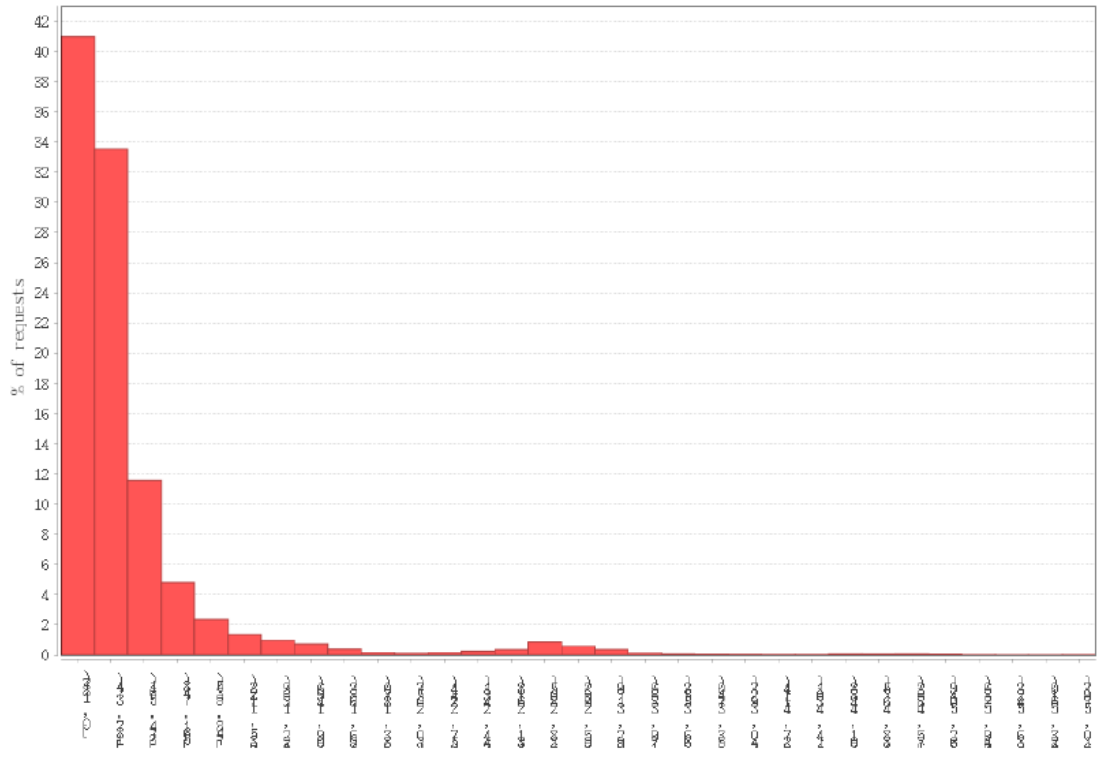
Average Trip Distance:	14.28 km

Wait Times

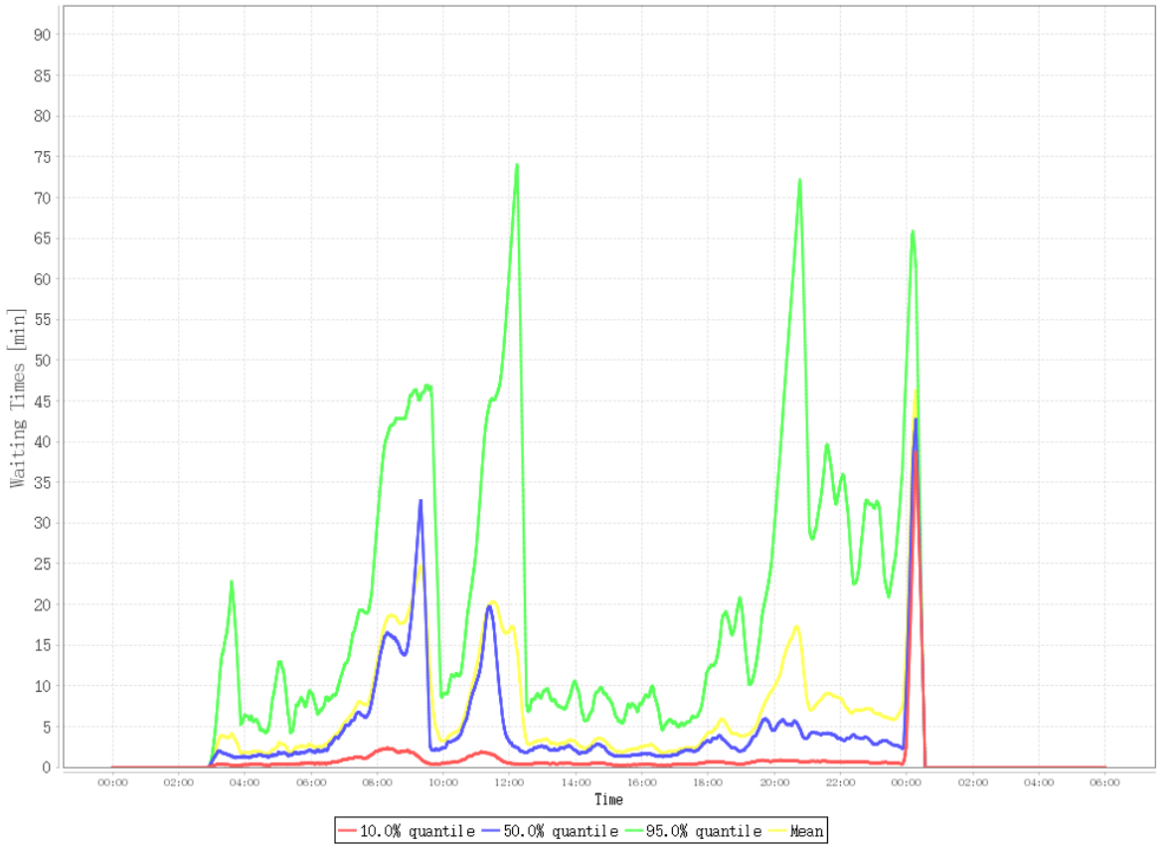
10.0% quantile	00:01:12
50.0% quantile	00:03:41
95.0% quantile	00:19:59
Mean	00:06:12
Maximum:	01:33:36

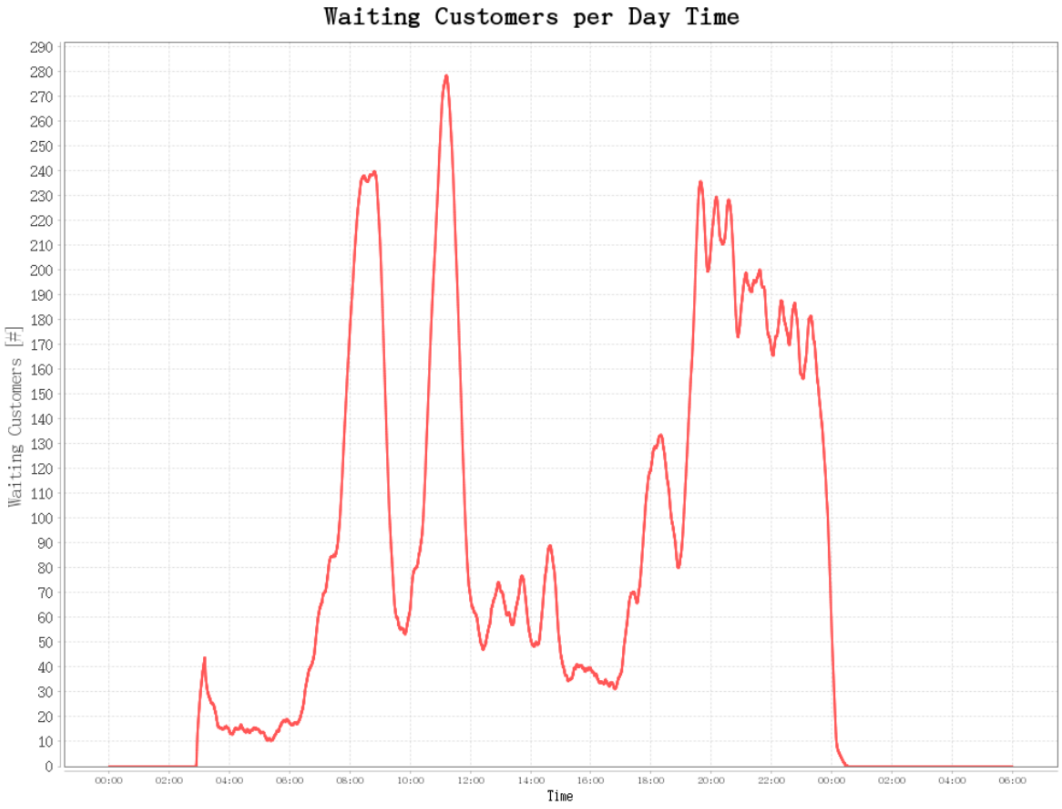


Number of Requests per Wait Time

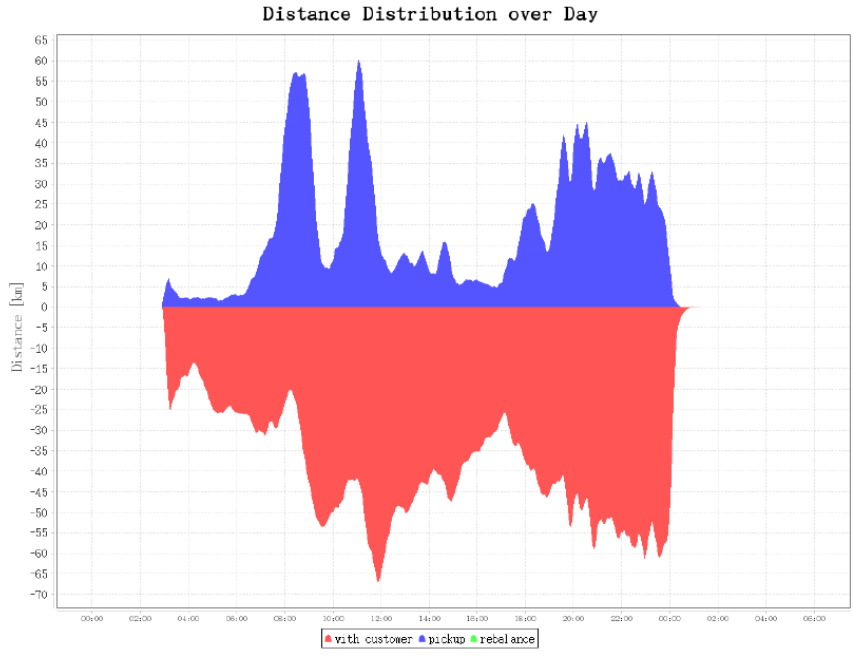


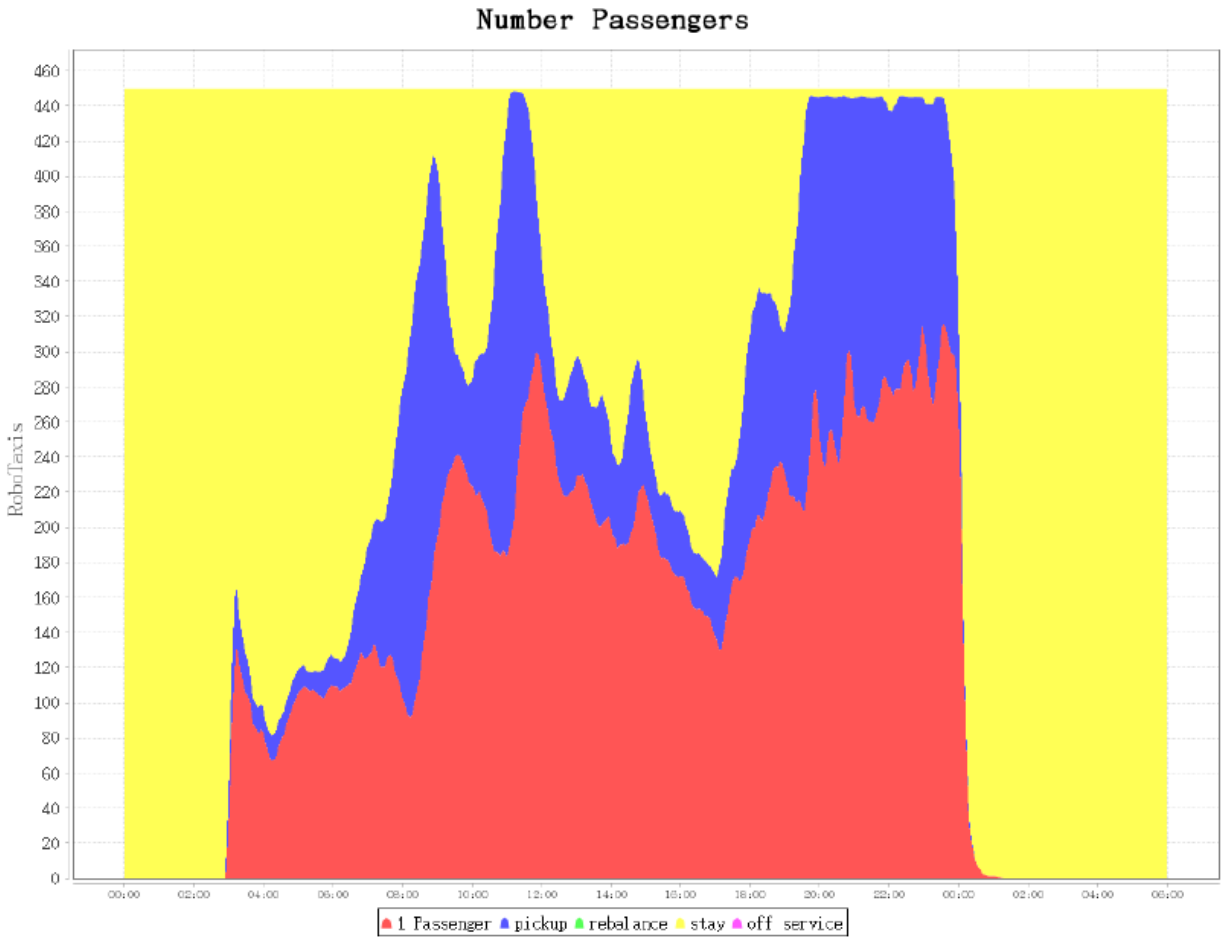
Binned Waiting Times





Fleet Efficiency




Number Passengers
RoboTaxis
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
380
400
420
440
460
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
1 Passenger
pickup
rebalance
stay
off service

SIMULATION 2

DISPATCH PERIOD : 15

NO OF VEHICLES: 500

User:	dwang
Timestamp:	2019/05/02 - 11:17:43
Iterations:	1 ,i.e., 0 in matsim config.

AV File AV_Config File

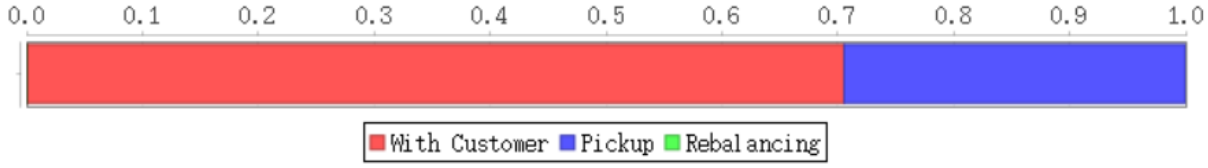
Dispatcher:	DemandSupplyBalancingDispatcher
Rebalancing Period:	-00:00:01
Redispatching Period:	00:00:15
Network:	null_prepared
Virtual Nodes:	10 virtual nodes.
Population:	21562
Number of Vehicles:	500
Number of Requests	21562
Distance Ratio:	70.55%
Occupancy Ratio:	27.35 %

Distances

Total:	436564.85 km
Rebalancing:	0.00 km (0.00%)
Pickup:	128572.18 km (29.45%)
With Customer:	307992.68 km (70.55%)

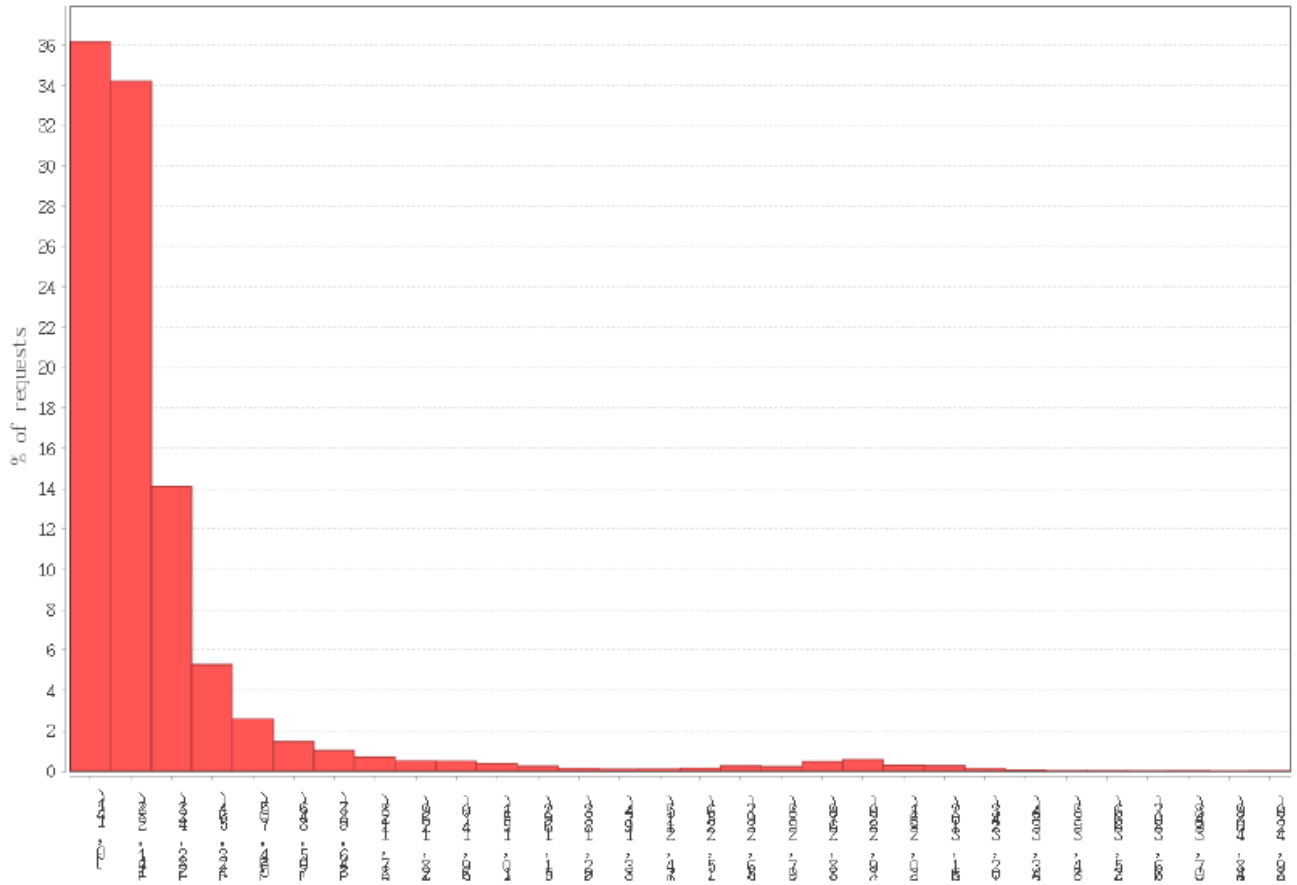
Average Trip Distance:	14.28 km

Total Distance Distribution

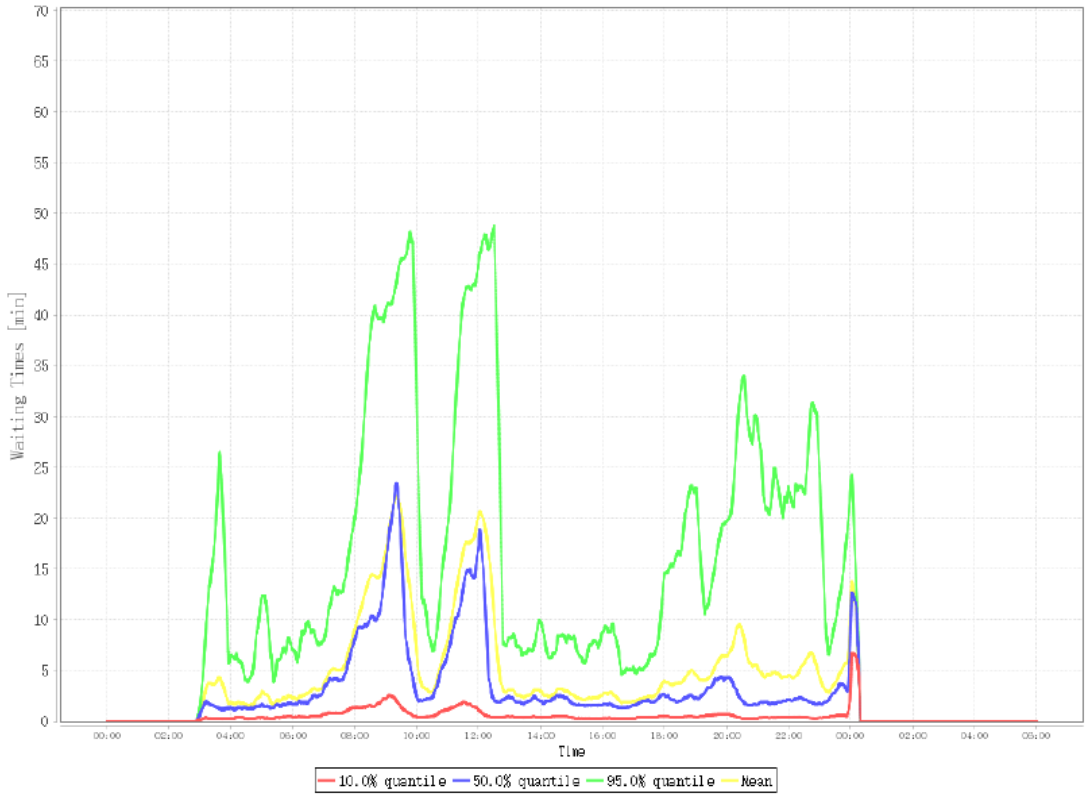


Wait Times	
10.0% quantile	00:00:57
50.0% quantile	00:03:06
95.0% quantile	00:17:07
Mean	00:05:13
Maximum:	01:10:16

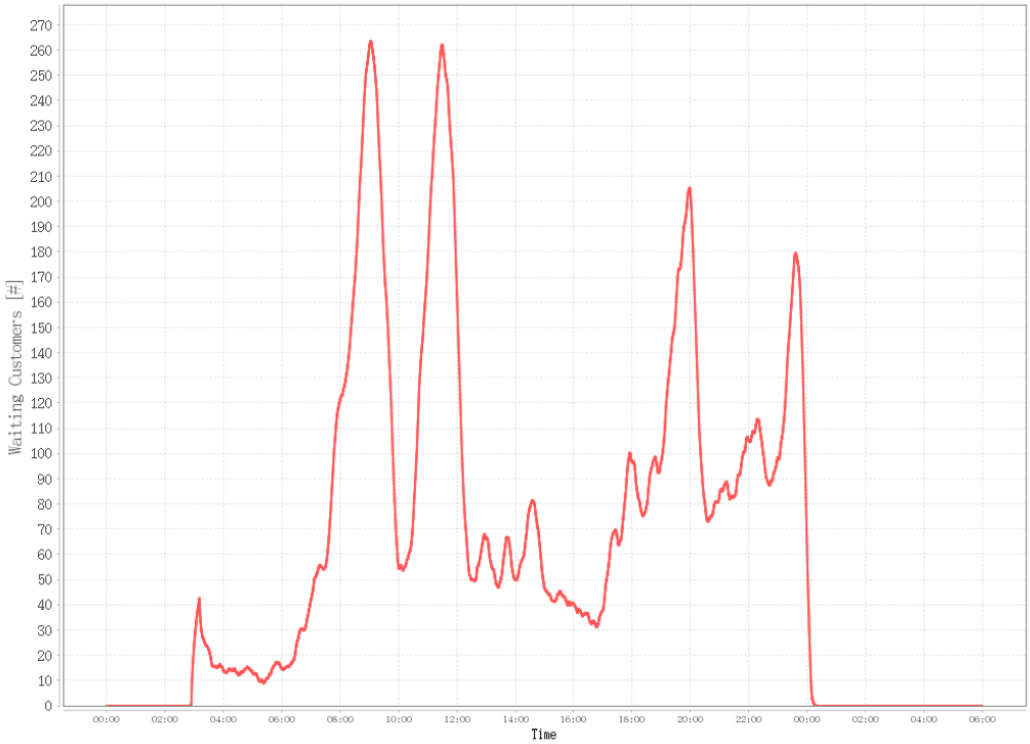
Number of Requests per Wait Time

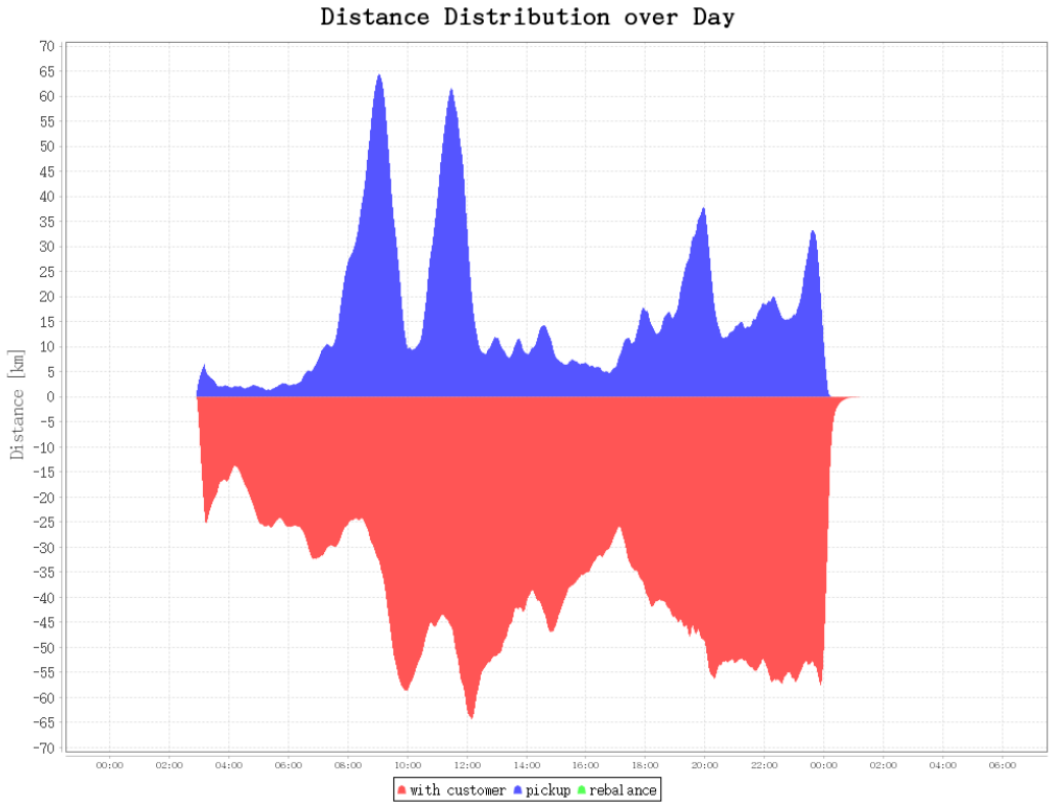


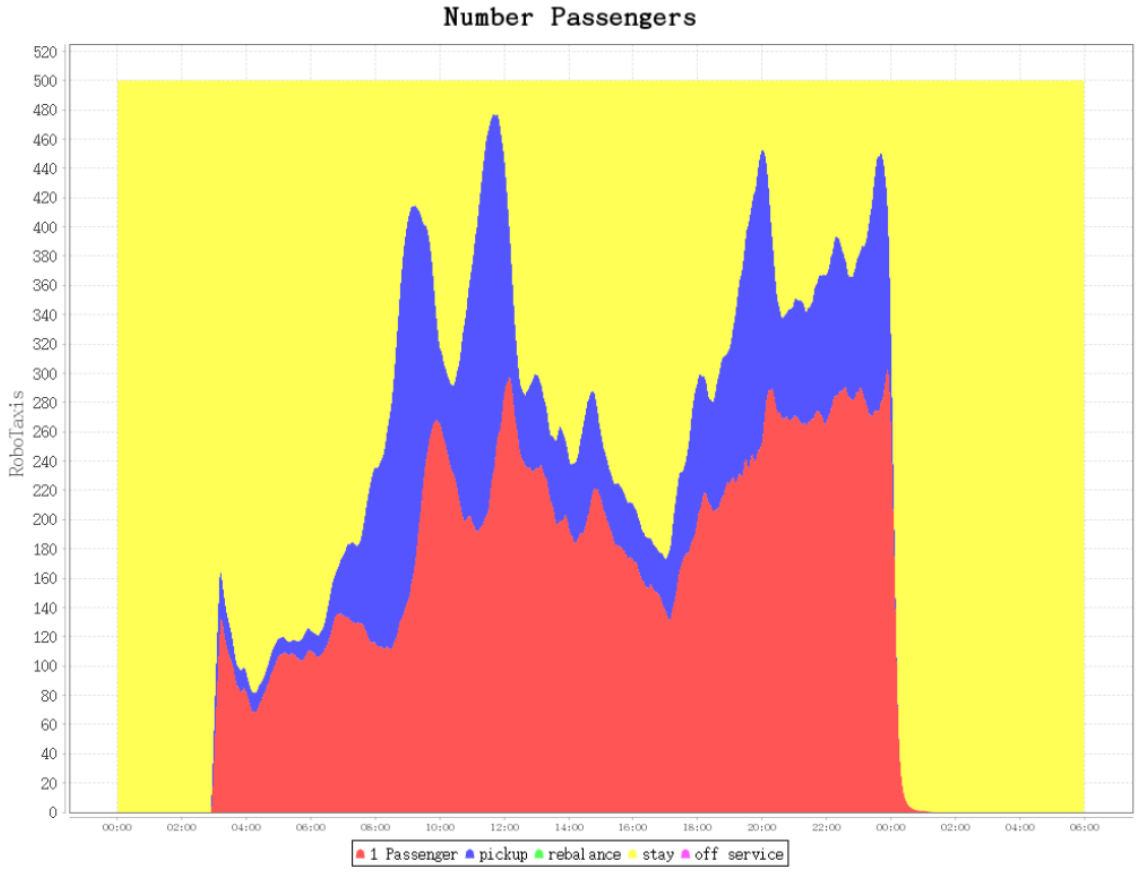
Binned Waiting Times



Waiting Customers per Day Time




Distance Distribution over Day
Distance [km]
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
-55
-60
-65
-70
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
with customer
pickup
rebalance


Number Passengers
RoboTaxis
520
500
480
460
440
420
400
380
360
340
320
300
280
260
240
220
200
180
160
140
120
100
80
60
40
20
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
1 Passenger
pickup
rebalance
stay
off service

SIMULATION 3

NO OF VEHICLES: 450

DISPATCH PERIOD: 10

User:	tl456
Timestamp:	2019/05/02 - 02:45:41
Iterations:	1 ,i.e., 0 in matsim config.

AV File AV_Config File

Dispatcher:	HighCapacityDispatcher
Rebalancing Period:	-00:00:01
Redispatching Period:	00:00:20
Network:	null_prepared
Virtual Nodes:	10 virtual nodes.
Population:	21562
Number of Vehicles:	500
Number of Requests	21562
Distance Ratio:	82.61%
Occupancy Ratio:	25.33 %

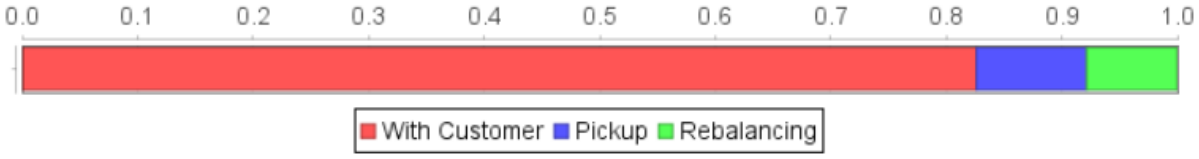
Distances

Total:	336932.79 km
Rebalancing:	26413.57 km (7.84%)
Pickup:	32184.77 km (9.55%)
With Customer:	278334.45 km (82.61%)
Average Trip Distance:	12.91 km

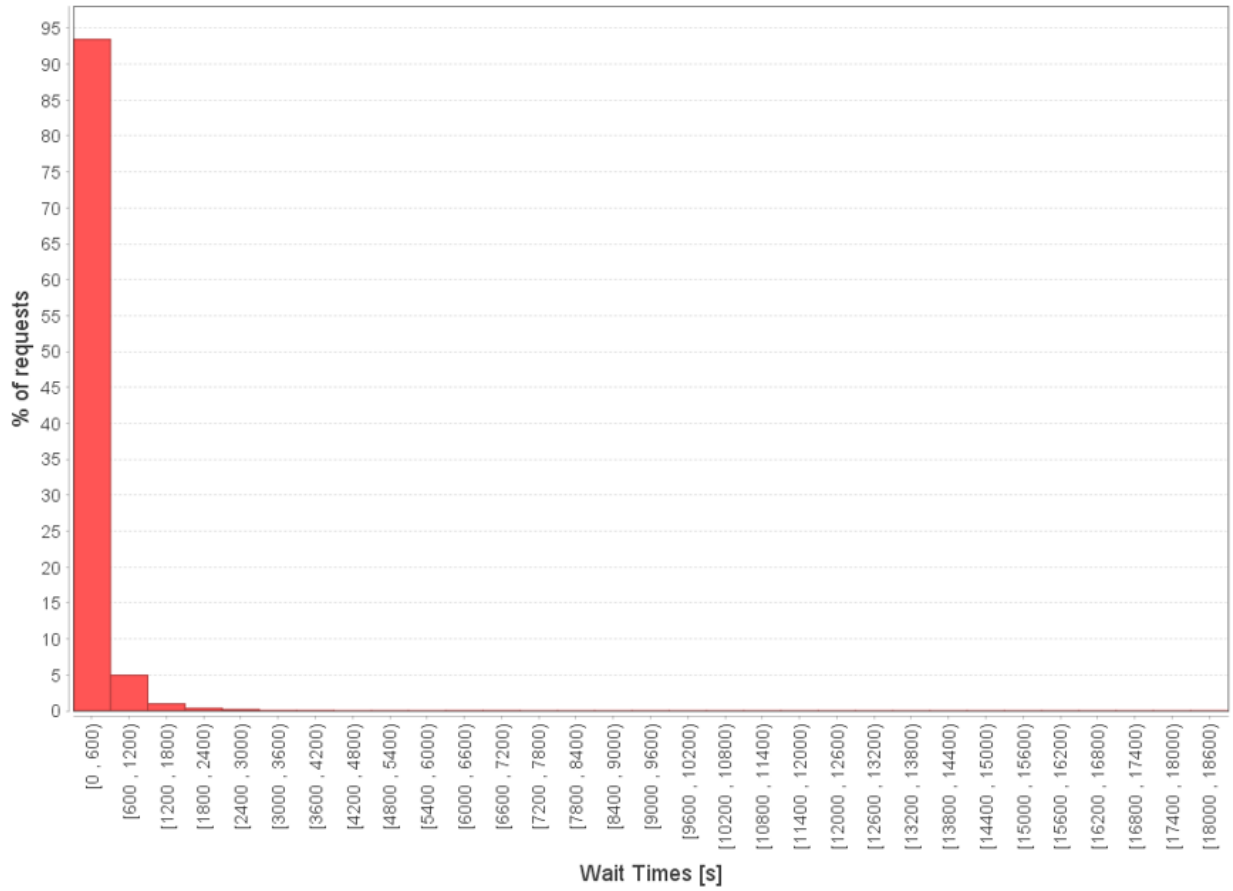
Wait Times

10.0% quantile	00:01:15
50.0% quantile	00:03:08
95.0% quantile	00:11:44
Mean	00:04:19
Maximum:	05:00:04

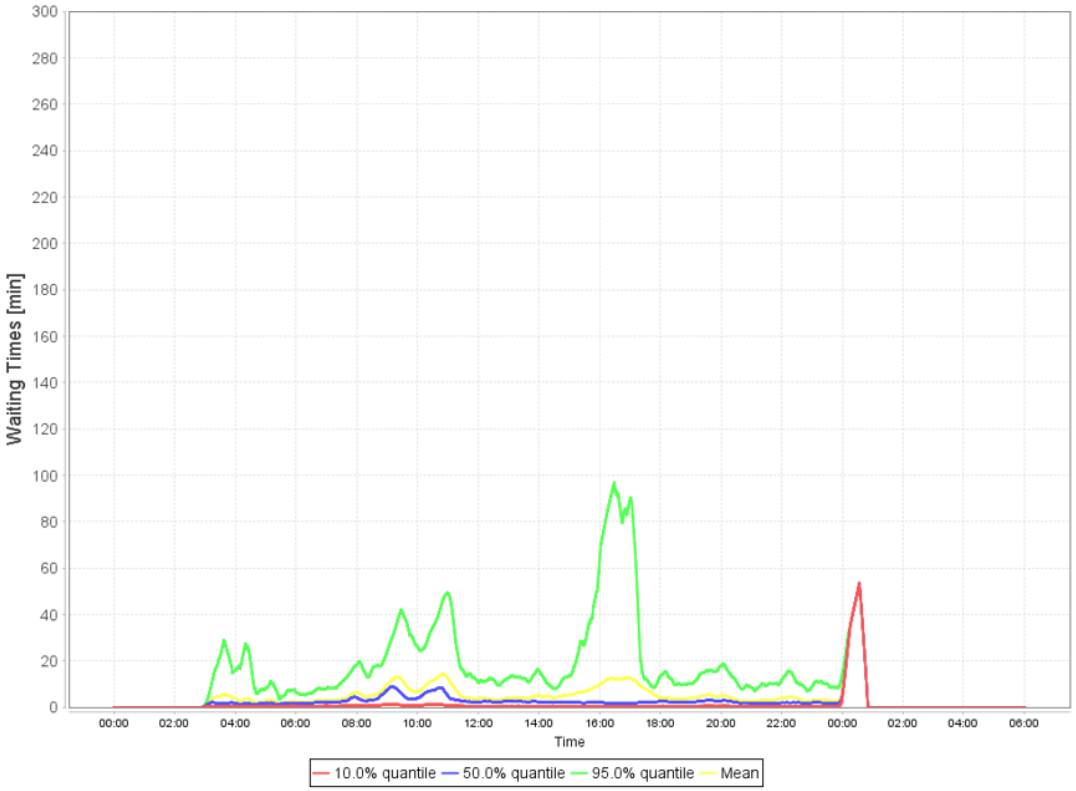
Total Distance Distribution



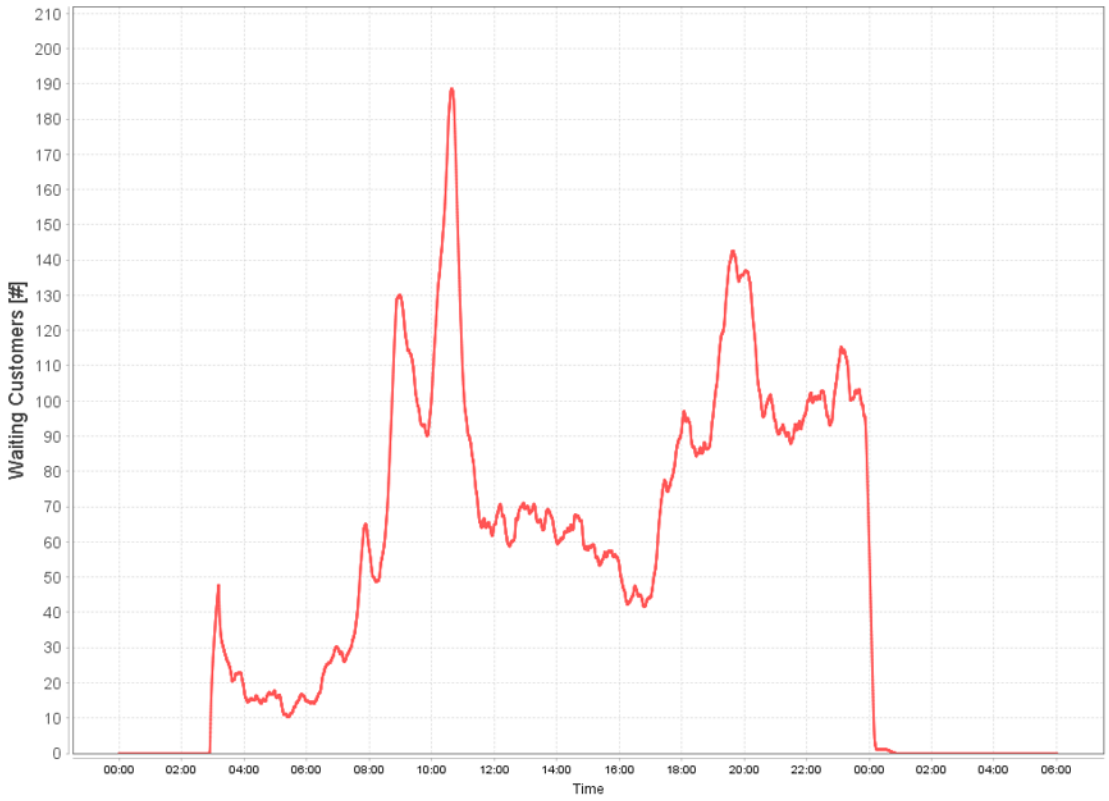
Number of Requests per Wait Time



Binned Waiting Times

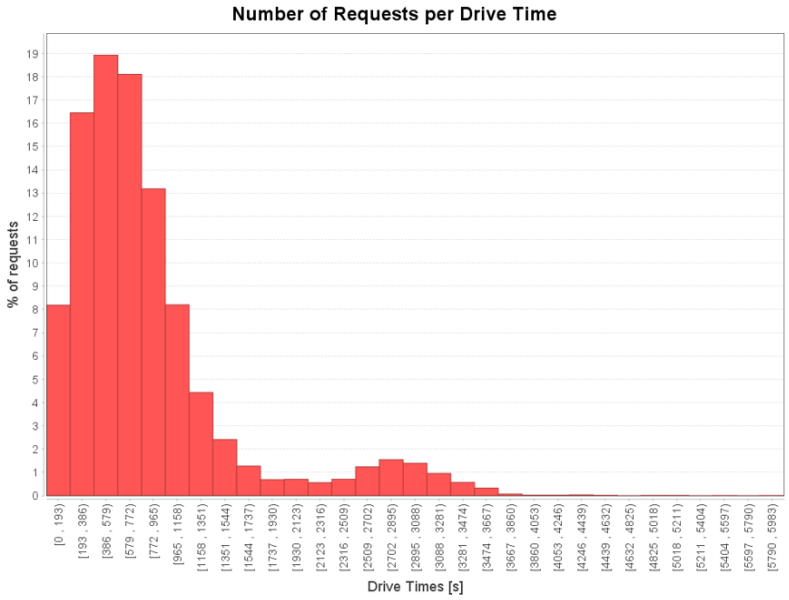


Waiting Customers per Day Time



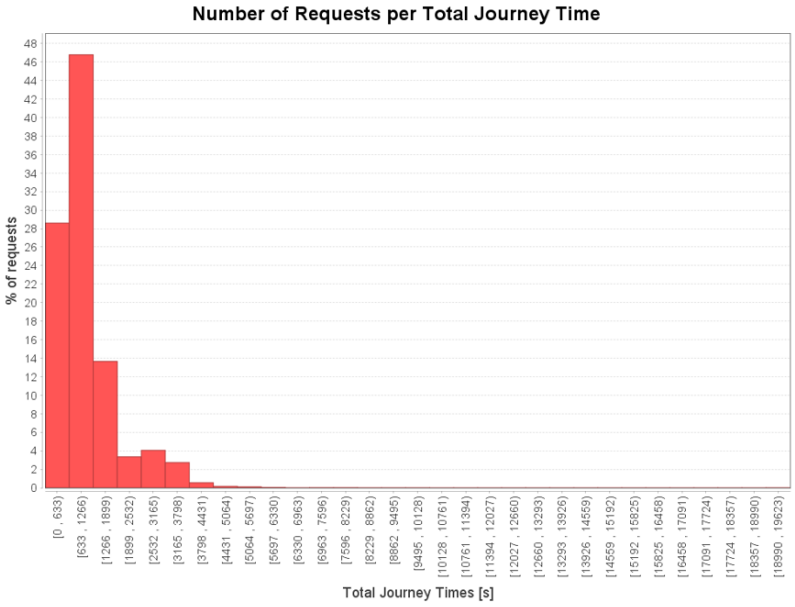
Drive Times

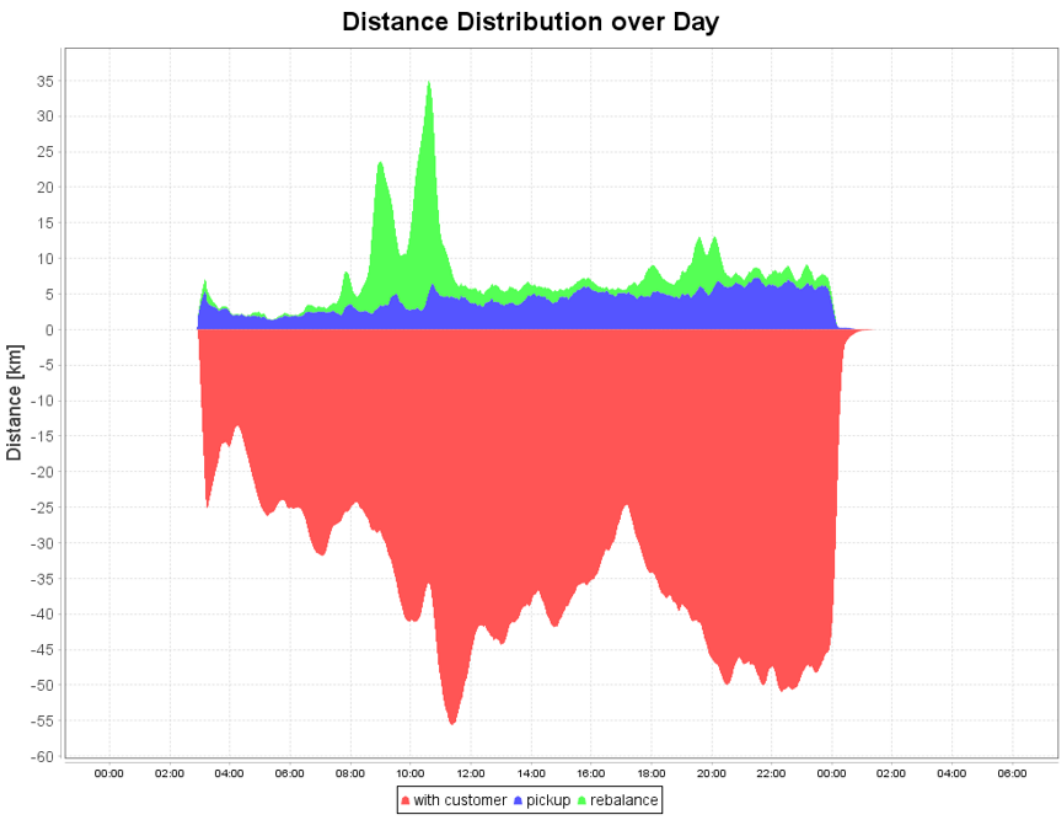
10.0% quantile	00:03:40
50.0% quantile	00:10:40
95.0% quantile	00:45:00
Mean	00:13:41
Maximum:	01:36:40

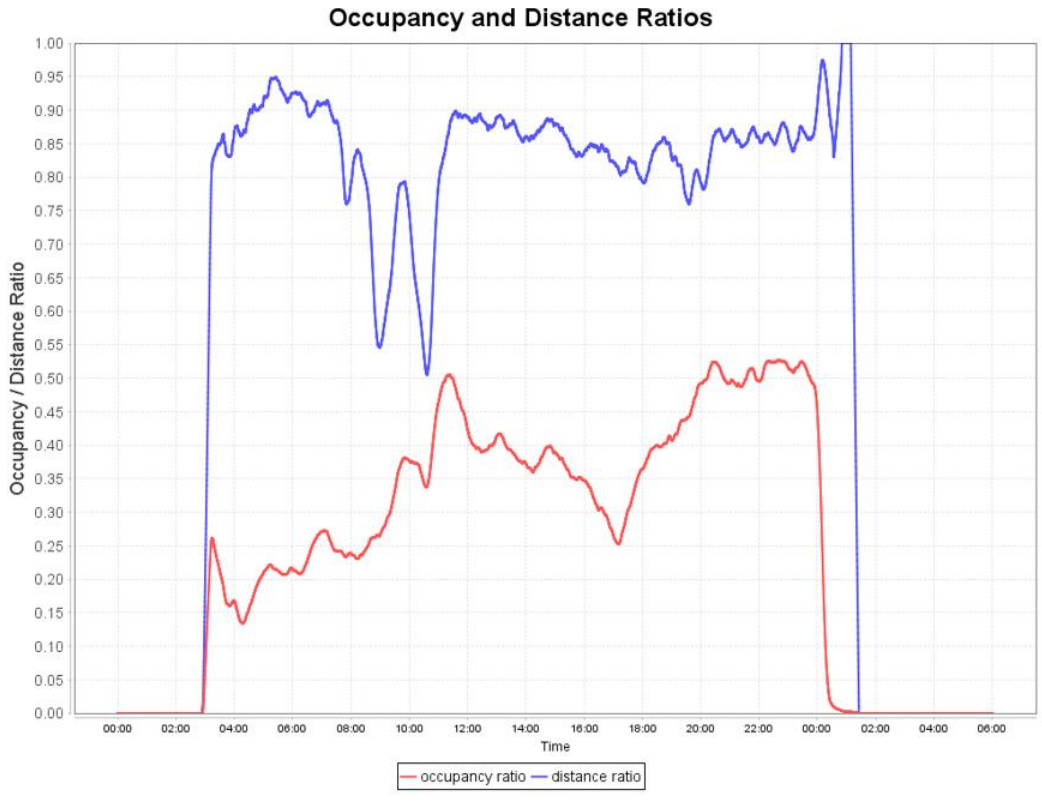


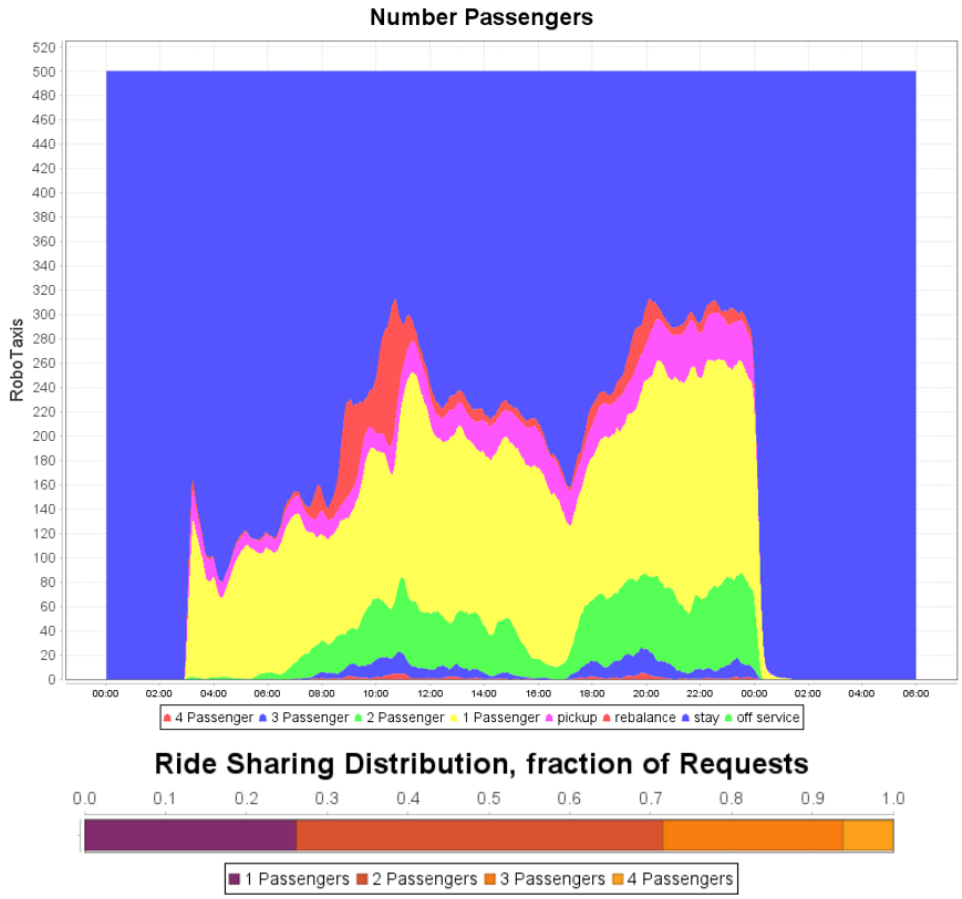
Total Journey Times

10.0% quantile	00:06:40
50.0% quantile	00:14:39
95.0% quantile	00:49:47
Mean	00:18:01
Maximum:	05:16:44




Distance Distribution over Day
Distance [km]
35
30
25
20
15
10
5
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
-55
-60
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
with customer
pickup
rebalance


Occupancy and Distance Ratios
Occupancy / Distance Ratio
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
Time
occupancy ratio
distance ratio



CONCLUSION

In Simulation 3 the mean wait times are the lowest with the parameters chosen. This gives the most efficient operation of the taxis.

Graphs Explained

1. Number of Requests per Wait Time: This graph shows that 94% of the requests have a wait time of less than 10mins and 99% of the requests have wait times under 20mins.
2. Waiting Customers per Day Time: Most of the customers were waiting between 10:00am to 12:00pm and 6:00pm to 8:00pm. This was due to high traffic going to work and coming back from work. It could be seen in the simulation.
3. Number of Requests per Drive: This graph showed the drive time of the cars. Most of the car were travelling for drive time 5-20mins. This can help companies in deciding the distribution of vehicles by location.
4. Number of Requests per Journey Time: This graph showed the total journey times of the customers. The mean journey time was 20mins.
5. Occupancy Ratio: This gives the ratio between taxi being occupied and unoccupied. The average occupancy was 50%.
6. Number Passengers: The no of passengers travelling by a taxi. Most of the robotaxis were occupied by 1 passenger.