TIA on SUMO- Existing Analysis

1.Site Selection and Inventory

User defines the radius of influence. Need a UI for it

Lists out all the major and minor roads and intersections. Stop controlled and signalized intersections.

Define and draw the development area

Build your network

map.net.xml

Draw area on OSM and download the maps. Upload that osm to create a network file on SUMO.

cd "C:\Users\nikit\Downloads\TIA_1"

netconvert ^

--osm-files map.osm ^

--output-file map.net.xml ^

--geometry.remove true ^

--geometry.remove.min-length 5 ^

--geometry.min-dist 2 ^

--geometry.max-segment-length 50 ^

--junctions.join true ^

--tls.guess true ^

--tls.guess-signals true ^

--sidewalks.guess ^

--crossings.guess ^

--walkingareas

What each important option does

- *--geometry.remove, --geometry.remove.min-length, --geometry.min-dist, --geometry.max-segment-length*
  Simplify edge shapes and remove short geometry nodes so the network is less jagged while keeping reasonable curvature. [sumo.dlr.de+1]

- *--junctions.join*
  Merges close nodes into a single junction cluster so intersections look and behave like proper junctions rather than multiple tiny nodes. sumo.dlr.de+1

- *--tls.guess, --tls.guess-signals*
  Let netconvert automatically create traffic light controllers and infer signal heads around OSM style intersections. sumo.dlr.de

- *--sidewalks.guess, --crossings.guess, --walkingareas*
  Guess sidewalks from road attributes and then generate pedestrian crossings and walking areas at junction corners, which you can then use for pedestrian simulation. sumo.dlr.de+1

After running this, open map.net.xml in NETEDIT, check a few key intersections, and manually fix anything that looks wrong such as odd channelization or missing approaches.

Network Builder Checker

Build a network, then run automated validation before accepting it. Use four checks:

1. Topology check
   - Ensure one main connected component
   - No dangling major roads
   - One–way and two–way directions correct
2. Turn-movement check
   - Read OSM turn-restriction relations
   - For each junction, ensure each approach has a legal exit
   - Remove illegal turns or missing U-turns
   - Validate left, through, right angles based on geometry
3. Geometry check
   - Detect unrealistic curves, kinks, very short edges
   - Confirm that roads actually meet at intersections
   - Validate divided roads (no direct turns across medians)
4. Routing and simulation check
   - Run sample OD routing and compare with OSRM/Valhalla
   - Run a light simulation; flag vehicles that fail to reach destination
   - Identify intersections where vehicles dead-end or loop

Use these checks as a pipeline after every network build. The agent only approves networks that pass all four.

Metrics

1. Percentage of nodes where at least one approach has no legal exit
2. Percentage of edges that are unreachable from the giant component
3. Fraction of sampled OD pairs where your route diverges significantly from an external router
4. Fraction of vehicles in diagnostic simulation that fail to reach their destination in time

---

2. List out intersections in the area of influence

1. Parse the net file
   Use the SUMO net xml (for example map.net.xml)
2. Filter only real intersections
   Keep junctions whose type is not "internal"
   Typical intersection types are "traffic_light", "priority", "right_before_left", "priority_stop"
3. For each such junction, collect
   • id
   • type
   • coordinates x and y
   • incoming lanes and edges

Run the list python file

3.Process volumes

Some additional factors need to be calculated for these tables. Peak hour and peak hour factor will be calculated for both AM and PM peak cycles.

Peak hour denotes to the time when 4 continuous 15 min volume data have the highest count during the peak period. The peak hour factor (**PHF**) is a ratio used in traffic engineering to measure the consistency of traffic flow within a peak hour. It is calculated by dividing the total traffic volume during the peak hour by the peak flow rate during the busiest 15-minute interval within that hour. A PHF of 1.0 indicates uniform flow, while a lower PHF signifies greater fluctuations in traffic during the hour.

1 Identify the peak hour window

For each period AM and PM do this separately

1.1 For each 15 minute row compute Total_veh
Sum all vehicle movements in that row
For example EB L plus EB T plus EB R plus WB L plus …

1.2 Compute the rolling sum of four consecutive rows
RollingHour[i] = Total_veh[i] plus Total_veh[i+1] plus Total_veh[i+2] plus Total_veh[i+3]

1.3 Find the index k where RollingHour[k] is maximum
Rows k to k+3 are your peak hour for that period
Store k for AM and k for PM

---

2 Compute peak hour volumes per movement

For each movement m (EB left EB through EB right etc) and for each period

2 1 Take the four rows in the peak hour window k to k+3

2 2 Hourly volume for movement m
V_hour(m) = v_m[k] plus v_m[k+1] plus v_m[k+2] plus v_m[k+3]

2 3 Store V_hour(m)
This is what you will feed into LOS tools and into SUMO if you want a flat peak hour demand

---

3 Compute PHF per movement

The standard definition

PHF for movement m

PHF(m) = V_hour(m) divided by (4 × V_15max(m))

where

V_15max(m) = max(v_m[k], v_m[k+1], v_m[k+2], v_m[k+3])

Since your volumes are 15 minute counts this is direct

You can also compute an overall intersection PHF using total volumes instead of per movement if you need that for reports

---

4 Extract pedestrians and trucks

Use from the Excel file directly

4.1 Use the same peak hour window k to k+3

4.2 For pedestrians on each crosswalk or leg

Ped_hour = ped[k] plus ped[k+1] plus ped[k+2] plus ped[k+3]

4.3 For trucks for each leg or movement

If trucks are given as separate counts

Trk_hour(m) = trk_m[k] plus trk_m[k+1] plus trk_m[k+2] plus trk_m[k+3]

Then heavy vehicle percent for movement m

HV_pct(m) = 100 × Trk_hour(m) divided by V_hour(m)

If trucks are given as percent directly in Excel, just carry that value through after checking it is consistent over the peak hour rows

---

5. Build the final data record per movement

For each intersection I and each movement m and each period P create a structured record like

- intersection_id
- period AM or PM
- approach EB WB NB SB
- movement L T R U
- V_hour
- PHF
- HV_pct
- Ped_hour

4. Mapping Intersections and Edges

Link catalog intersections to your TMC or count sheet ids

1. Take your turning movement count data in a structured table
   columns such as intersection name or id, leg, movement, period AM or PM, volume.
2. For each count location, match it to a junction in the catalog by
   a. matching street names from the table to street names on approaches
   b. confirming the number of legs matches
   c. checking that approach directions roughly match (N S E W).
3. Persist a mapping
   example
   count_intersection_id → **junction_id** in SUMO

This step can be interactive in a simple web or CLI UI where the agent shows candidate junctions and you pick one.

For each mapped intersection

1. Take the approaches from your script output
   example Eastbound on Woodmen, Westbound on Chambers, etc.
2. For each approach define
   a. approach edge id (incoming edge to junction)
   b. outgoing edge ids for each movement L T R U.
3. Create a small mapping structure such as

(intersection_id, approach_name, movement) → (from_edge_id, to_edge_id)

You can infer movement type by comparing angles between from and to edges, or allow a once off manual configuration per intersection that you then save.

Example (you choose based on your map.net.xml):

- EB approach in SUMO: E_in
    - EB Left → from="E_in" to to="N_out"
    - EB Thru → from="E_in" to to="W_out"
    - EB Right → from="E_in" to to="S_out"

Similarly for WB, NB, SB.

This mapping is static and can be stored as:

(leg, movement) → (from_edge, to_edge)

e.g. ("EB", "Thru") → ("E_in", "W_out")

5.1

Format volumes as per UTDF Format(In case of Synchro API)

Upload traffic volume in the following format. The traffic tables from the data collection consist of turning movement counts, mainlane counts , truck traffic and extra information on the TMC. The extra information is on pedestrians who could conflict with the turning movement.
The interface reads and writes CSV files compliant with the UTDF (Universal Traffic Data Format) format defined by Trafficware.

UTDF VOLUME
Volume data compatible with Synchro, Visum, and Vistro software.
only supports 15 or 60 minute intervals, enter study ID and select interval to export

Turning Movement Count																		
15 Minute Counts																		
DATE	TIME	INTID	SBR	SBT	SBL	SBU	WBR	WBT	WBL	WBU	NBR	NBT	NBL	NBU	EBR	EBT	EBL	EBU
21/05/2009	1200	181682	45	236	86	0	53	135	104	0	76	137	18	1	36	216	71	0
21/05/2009	1215	181682	52	211	79	0	58	170	108	0	80	158	26	0	32	205	65	0
21/05/2009	1230	181682	41	189	70	0	46	141	105	1	94	171	28	3	31	161	68	0
21/05/2009	1245	181682	38	185	71	1	55	166	87	0	80	174	23	2	17	219	78	0

UTDF LANE

Lane data compatible with Synchro, Visum, and Vistro software.
enter study ID and select interval to export

Lane Group Data																	
RECORDNAME	INTID	SBR	SBT	SBL	SBU	WBR	WBT	WBL	WBU	NBR	NBT	NBL	NBU	EBR	EBT	EBL	EBU
Volume	181682	176	821	306	1	212	612	404	1	330	640	95	6	116	801	282	0
Peds	181682	42	0	14	0	8	0	23	0	14	0	42	0	23	0	8	0
PHF	181682	0.73	0.75	0.69	0.08	0.77	0.78	0.67	0.08	0.76	0.74	0.66	0.25	0.64	0.75	0.73	0
HeavyVehicles	181682	2	2	1	0	3	3	5	0	6	3	3	0	2	4	3	0

Naming conventions for signal control parameters

The following table shows the relation between the names used in Aimsun Next and those used in the UTDF file for signal control parameters.

Naming conventions for NEMA parameters:

Aimsun Next UTDF ---------- ----------

Barrier, Ring, Position BRP

Minimum Green, Maximum Initial Green MinGreen

Max-Out MaxGreen

Passage Time VehExt

Time Before Reduce TimeBeforeReduce

Time To Reduce TimeToReduce

Passage Time MinGap

Yellow Time Yellow

Interphase Time AllRed

Recall Recall

Permissive Period From, *Permissive Period to* and *Force Off* are initialized automatically.

Sources:

1. https://docs.aimsun.com/next/22.0.1/UsersManual/SynchroImporter.html
2. https://help.miovision.com/s/article/Report-Types-in-Traffic-Data-Online?language=en_US

5.1 Format volumes as per SUMO format

At this step you have three core tables (they can be CSV, database tables, or data frames).

Intersections
• intersection_id
• sumo_node_id
• control type (optional)

Approaches and movements
• intersection_id
• approach_id
• from_edge_id (SUMO)
• to_edge_id (SUMO)
• movement_type (through, left, right)

Volume table
• intersection_id
• approach_id
• period (AM, PM)
• interval_start, interval_end in seconds relative to period
• movement_type
• volume_veh (fifteen minute total)

5.2 Assign volumes as per SUMO format

This is a detailed, step-by-step breakdown of how to perform Trip Distribution and Trip Assignment in SUMO specifically when you possess Turning Movement Counts (TMCs) for AM and PM peak periods.

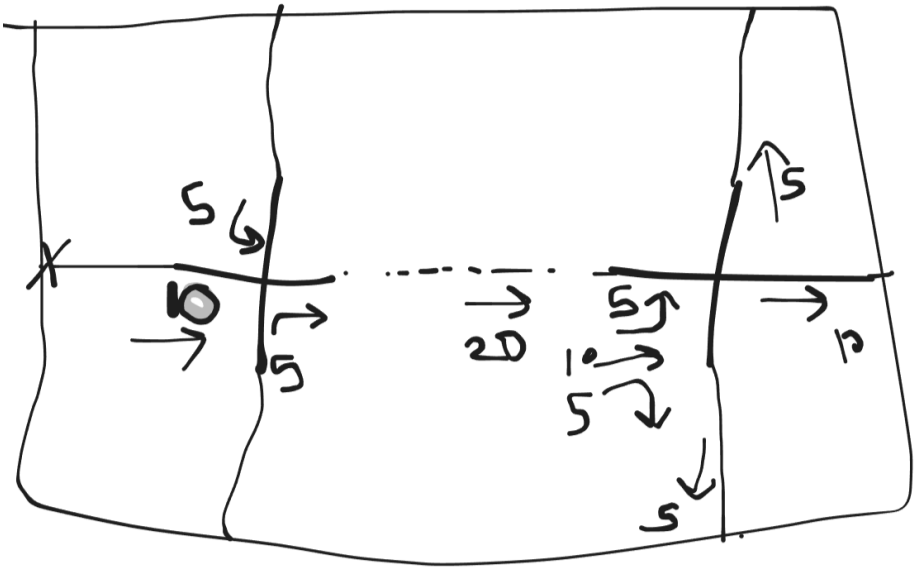
In the context of SUMO, we do not use the traditional "Gravity Model" (Distribution) or "User Equilibrium" (Assignment) when we have specific intersection counts. Instead, we use a method called Junction Turning Ratio Routing.

Here is the detailed workflow.

Phase 1: Conceptual Framework

Before writing code, you must understand how SUMO translates TIA concepts:
Trip Distribution becomes "Turn Probabilities". instead of saying "100 trips go from Zone A to Zone B," we say "Of the 500 cars approaching Intersection 1, 20% turn right and 80% go straight."

Trip Assignment becomes "Flow Routing (jtrrouter)". SUMO generates individual vehicles at the starting point and "rolls the dice" at every intersection based on your probabilities to decide their path.



Phase 2: Data Preparation (The Math)

You have raw volumes (e.g., 500 vehicles/hour). SUMO requires probabilities (0.0 to 1.0). You must process your AM and PM data separately.

Step 2.1: Calculate Approach Totals

For every approach at every intersection, sum the movements.

- Example: Intersection "Main_St_&_1st_Ave" (AM Peak)
    - Left Turn Volume: 100
    - Through Volume: 800
    - Right Turn Volume: 100
    - Total Approach Volume: 1000 vehicles

Step 2.2: Calculate Turn Ratios (Probabilities)

Divide each movement by the total approach volume.

- $P(Left) = 100 / 1000 = 0.10$
- $P(Thru) = 800 / 1000 = 0.80$
- $P(Right) = 100 / 1000 = 0.10$

Note: The sum must always equal 1.0 (100%). If it does not, SUMO will throw an error or normalize it automatically, which might skew your data.

Phase 3: Creating the "Flow" Files (Trip Generation)

In this step, we define the **Sources**. These are the edges where traffic *enters* your simulation network. You typically only define flows for the boundary edges of your network.

You need two files: flows_AM.xml and flows_PM.xml.

```
<routes>
    <vType id="car" accel="2.6" decel="4.5" sigma="0.5" length="5" minGap="2.5" maxSpeed="55.55"/>

    <flow id="flow_West_AM" begin="0" end="3600" number="1000" from="Edge_West_In" type="car"/>

    <flow id="flow_North_AM" begin="0" end="3600" number="450" from="Edge_North_In" type="car"/>

</routes>
```

Detailed Explanation of Attributes:

- **id**: A unique name for this stream of traffic.
- **begin="0" end="3600"**: This defines your Peak Hour. 0 to 3600 seconds = 1 Hour.

- **number="1000"**: This tells SUMO to inject 1,000 cars evenly distributed over that hour.
- **from**: The **Edge ID** where the cars appear. This must be the incoming lane at your map boundary.

Phase 4: Creating the "Turn Ratio" Files (Trip Distribution)

This is the most labor-intensive part. You must map the probabilities calculated in Phase 2 to the specific **From-Edge** and **To-Edge** IDs in your network.

You need two files: turns_AM.xml and turns_PM.xml.

```
<turns>
   <interval begin="0" end="3600">

      <fromEdge id="Edge_West_In">
         <toEdge id="Edge_North_Out" probability="0.10"/>

         <toEdge id="Edge_East_Out" probability="0.80"/>

         <toEdge id="Edge_South_Out" probability="0.10"/>
      </fromEdge>

      <fromEdge id="Edge_Next_In">
          <toEdge id="Edge_Next_Out" probability="1.0"/>
      </fromEdge>

   </interval>
</turns>
```

Crucial Details:

1. **Incoming Edge (fromEdge):** This is the edge *approaching* the traffic light.
2. **Outgoing Edge (toEdge):** This is the edge *departing* the traffic light.
3. **Missing Turns:** If you forget to define an edge, SUMO will assume the car continues straight or chooses a random path.

Phase 5: Trip Assignment (Running jtrrouter)

Now we perform the Assignment. In traditional modeling, "Assignment" means finding the path of least resistance (User Equilibrium).

In this TIA workflow using jtrrouter, Assignment means: *"Generate a specific list of edges (a route) for every single car such that the aggregate counts match the Turn Ratios file."*

You must run this command twice (once for AM, once for PM).

```
jtrrouter --net-file your_network.net.xml \
    --route-files flows_AM.xml \
    --turn-ratio-files turns_AM.xml \
    --output-file traffic_AM.rou.xml \
    --begin 0 \
    --end 3600 \
    --accept-all-destinations
```

Detailed Breakdown of Flags:

- **--route-files**: Input the flows.xml (The Source volumes).
- **--turn-ratio-files**: Input the turns.xml (The Intersection percentages).
- **--output-file**: This is the result. It will contain the generated trips.
- **--accept-all-destinations**: **This is the most important flag for TIA.**
    - *Why?* In a Turning Count model, we don't know the final destination of the car. We just know it turns right, then goes straight, then turns left. Eventually, it will hit the edge of your map.
    - Without this flag, SUMO will delete the car because "it didn't reach a valid destination."
    - With this flag, SUMO says, "The car hit a dead-end edge (boundary). I will assume this was its destination and let it exit gracefully."

This is a detailed, step-by-step breakdown of how to perform **Trip Distribution** and **Trip Assignment** in SUMO specifically when you possess **Turning Movement Counts (TMCs)** for AM and PM peak periods.
In the context of SUMO, we do not use the traditional "Gravity Model" (Distribution) or "User Equilibrium" (Assignment) when we have specific intersection counts. Instead, we use a method called **Junction Turning Ratio Routing**.
Here is the detailed workflow.

Phase 1: Conceptual Framework
Before writing code, you must understand how SUMO translates TIA concepts:

1. **Trip Distribution** becomes **"Turn Probabilities"**. instead of saying "100 trips go from Zone A to Zone B," we say "Of the 500 cars approaching Intersection 1, 20% turn right and 80% go straight."
2. **Trip Assignment** becomes **"Flow Routing (jtrrouter)"**. SUMO generates individual vehicles at the starting point and "rolls the dice" at every intersection based on your probabilities to decide their path.

Phase 2: Data Preparation (The Math)
You have raw volumes (e.g., 500 vehicles/hour). SUMO requires **probabilities** (0.0 to 1.0). You must process your AM and PM data separately.
Step 2.1: Calculate Approach Totals
For every approach at every intersection, sum the movements.

- **Example: Intersection "Main_St_&_1st_Ave" (AM Peak)**
    - Left Turn Volume: 100
    - Through Volume: 800
    - Right Turn Volume: 100
    - **Total Approach Volume:** 1000 vehicles

Step 2.2: Calculate Turn Ratios (Probabilities)
Divide each movement by the total approach volume.

- $P(Left) = 100 / 1000 = 0.10$
- $P(Thru) = 800 / 1000 = 0.80$
- $P(Right) = 100 / 1000 = 0.10$

Note: The sum must always equal 1.0 (100%). If it does not, SUMO will throw an error or normalize it automatically, which might skew your data.

Phase 3: Creating the "Flow" Files (Trip Generation)

In this step, we define the **Sources**. These are the edges where traffic *enters* your simulation network. You typically only define flows for the boundary edges of your network.

You need two files: flows_AM.xml and flows_PM.xml.

The XML Structure

XML

```
<routes>
   <vType id="car" accel="2.6" decel="4.5" sigma="0.5" length="5" minGap="2.5" maxSpeed="55.55"/>

   <flow id="flow_West_AM" begin="0" end="3600" number="1000" from="Edge_West_In" type="car"/>

   <flow id="flow_North_AM" begin="0" end="3600" number="450" from="Edge_North_In" type="car"/>

</routes>
```

Detailed Explanation of Attributes:

- **id**: A unique name for this stream of traffic.
- **begin="0" end="3600"**: This defines your Peak Hour. 0 to 3600 seconds = 1 Hour.
- **number="1000"**: This tells SUMO to inject 1,000 cars evenly distributed over that hour.
- **from**: The **Edge ID** where the cars appear. This must be the incoming lane at your map boundary.

Phase 4: Creating the "Turn Ratio" Files (Trip Distribution)

This is the most labor-intensive part. You must map the probabilities calculated in Phase 2 to the specific **From-Edge** and **To-Edge** IDs in your network.

You need two files: turns_AM.xml and turns_PM.xml.

The XML Structure

XML

```
<turns>
   <interval begin="0" end="3600">

      <fromEdge id="Edge_West_In">
         <toEdge id="Edge_North_Out" probability="0.10"/>

         <toEdge id="Edge_East_Out" probability="0.80"/>

         <toEdge id="Edge_South_Out" probability="0.10"/>
      </fromEdge>

      <fromEdge id="Edge_Next_In">
          <toEdge id="Edge_Next_Out" probability="1.0"/>
      </fromEdge>

   </interval>
</turns>
```

Crucial Details:

1. **Incoming Edge (fromEdge):** This is the edge *approaching* the traffic light.
2. **Outgoing Edge (toEdge):** This is the edge *departing* the traffic light.
3. **Missing Turns:** If you forget to define an edge, SUMO will assume the car continues straight or chooses a random path.

Phase 5: Trip Assignment (Running jtrrouter)

Now we perform the **Assignment**. In traditional modeling, "Assignment" means finding the path of least resistance (User Equilibrium).

In this TIA workflow using jtrrouter, Assignment means:
"Generate a specific list of edges (a route) for every single car such that the aggregate counts match the Turn Ratios file."
You must run this command twice (once for AM, once for PM).

The Command Structure

Bash

```
jtrrouter --net-file your_network.net.xml \
        --route-files flows_AM.xml \
        --turn-ratio-files turns_AM.xml \
        --output-file traffic_AM.rou.xml \
        --begin 0 \
        --end 3600 \
        --accept-all-destinations
```

Detailed Breakdown of Flags:

- **--route-files**: Input the flows.xml (The Source volumes).
- **--turn-ratio-files**: Input the turns.xml (The Intersection percentages).
- **--output-file**: This is the result. It will contain the generated trips.
- **--accept-all-destinations**: **This is the most important flag for TIA.**
    - *Why?* In a Turning Count model, we don't know the final destination of the car. We just know it turns right, then goes straight, then turns left. Eventually, it will hit the edge of your map.
    - Without this flag, SUMO will delete the car because "it didn't reach a valid destination."
    - With this flag, SUMO says, "The car hit a dead-end edge (boundary). I will assume this was its destination and let it exit gracefully."

Phase 6: The Result (The .rou.xml file)

After running the command in Phase 5, open the resulting traffic_AM.rou.xml. You will see something like this:

```
<vehicle id="flow_West_AM.0" depart="0.00">
   <route edges="Edge_West_In Edge_East_Out Edge_City_Link Edge_Exit"/>
</vehicle>
<vehicle id="flow_West_AM.1" depart="3.60">
   <route edges="Edge_West_In Edge_North_Out Edge_Suburb_Link Edge_Exit_2"/>
</vehicle>
```

Interpretation:

- SUMO has successfully Assigned a specific path to Vehicle 0.
- It calculated that based on your 80% Thru probability, this car should go East.
- Vehicle 1, however, "rolled the dice" and hit the 10% probability, so it was assigned the North path.
-

Phase 7: Handling Sinks (The Simulation Boundary)

There is one potential point of failure. If you have "Sink" edges (edges that leave the network) that are NOT explicitly marked as dead-ends in your .net.xml file, vehicles might stop at the end of the road and block traffic because they don't know they are allowed to vanish.

How to fix this in detailed steps:

1. Identify Sink Edges: List all edges that leave your map.

2. Create a Sink Definitions File (sinks.xml):

```
<additional>
    <edge id="Edge_Exit_1" type="sink"/>
    <edge id="Edge_Exit_2" type="sink"/>
</additional>
```

Include this in jtrrouter: Add --additional-files sinks.xml to your command in Phase 5. This forces jtrrouter to recognize these edges as valid simulation endings.

Example Scenario- just running random trips(big_trips.trips), routes(big_routes.rou) to get link wise results(tripinfo), configuration file is sim.sumocfg

Generate trips (Example)

```
cd "C:\Users\nikit\Downloads\TIA_1"
"C:\Users\nikit\AppData\Local\Programs\Python\Python313\python.exe" ^
 "C:\Program Files (x86)\Eclipse\Sumo\tools\randomTrips.py" ^
 -n "C:\Users\nikit\Downloads\TIA_1\map.net.xml" ^
 -o "C:\Users\nikit\Downloads\TIA_1\big_trips.trips.xml" ^
 -b 0 ^
 -e 3600 ^
 -p 0.5 ^
 --seed 15
```

Convert trips to full routes (Example)

```
C:\Users\nikit\Downloads\TIA_1>duarouter ^
More?   -n "C:\Users\nikit\Downloads\TIA_1\map.net.xml" ^
More?   -r "C:\Users\nikit\Downloads\TIA_1\small_trips.trips.xml" ^
More?   -o "C:\Users\nikit\Downloads\TIA_1\small_routes.rou.xml"
```

3. Run the config file (This is as an example run)

```
<configuration>
    <input>
        <net-file value="map.net.xml"/>
        <route-files value="routes_AM.rou.xml"/>
        <additional-files value="traffic_sensors.add.xml"/>
    </input>

    <output>
        <tripinfo-output value="tripinfo.xml"/>
```

```
    <queue-output value="queue.xml"/>

    <statistic-output value="summary.xml"/>
  </output>

  <time>
    <begin value="0"/>
    <end value="3600"/>
  </time>
</configuration>
```

In case demand doesn't run, run simulation

cd C:\Users\nikit\OneDrive\Traffic_Planning\TIA\Workflow\TIA_1

sumo-gui -n map.net.xml -r big_routes.rou.xml --begin 0 --end 3600

get simulation results

sumo -c sim.sumocfg

- **What happens:** SUMO runs silently. When it finishes, **tripinfo.xml**, **queue.xml**, and **summary.xml** will magically appear in your folder.
- **Note:** Your traffic_sensors.add.xml (which you included in the <input> section) will automatically generate the **edge_performance.xml** file as well.

File Name	Generated By	What TIA Result it contains
`tripinfo.xml`	`<tripinfo-output>`	**Delay (sec)** per vehicle. Used to calculate Intersection LOS.
`edge_performance.xml`	`traffic_sensors.add.xml`	**Volume (vph)** and **Speed**. Used to calibrate against your Counts.
`queue.xml`	`<queue-output>`	**Max Queue (meters)**. Used to check if turn lanes are overflowing.
`summary.xml`	`<statistic-output>`	**Total Vehicles**. A quick check (e.g., "Did all 1000 cars get through?").

6.2 Assign and Upload volumes (Using Synchro) Not Included in SUMO workflow

Two ways- UTDF Format or Streetlight API

Streetlight API Workflow

You can use the StreetLight Advanced Traffic Counts API in a workflow with the following steps:

Check available dates: Confirm what data is available for your travel mode.
Define your geometry: Provide the road segments you want to analyze.
Estimate segment count: Understand how many StreetLight segments your geometry includes.
Get segment geometry: Retrieve GeoJSON line strings for each segment.
Request metrics: Query metrics for your geometry with optional filters.

Source: https://developer.streetlightdata.com/reference/satc-segmentcount

Streetlight API Response

['segment_id', 'year_month', 'day_type', 'trips_volume']
[1000000000000001, '2020-10-01', 2, 2233]
[2000000000000001, '2020-10-01', 2, 4333]
[3000000000000001, '2020-10-01', 2, 10408]
[4000000000000001, '2020-10-01', 2, 1740]
[5000000000000001, '2020-10-01', 2, 10566]

7. Run Scenarios (No Build Existing and No Build Growth)

Now create two scenarios to simulate. Existing No Build mean present scenario with no development. Existing No Build and Growth mean grow volumes at a percentage and run simulation with future traffic. Once the volumes are fed, run the simulations to measure the existing intersections and segments in their present conditions. The performance of these intersections are measured using key performance metrics, also commonly called KPIs.
Synchro provides following performance metrics as output

A credible TIA must describe **operational performance** at each critical intersection and corridor. At minimum, you need:

A. Intersection Performance Metrics
These are mandatory for existing conditions:
1. Control Delay (seconds per vehicle)
Most critical metric.
Used to compute LOS per HCM tables.
2. Level of Service (LOS A–F)
Based on control delay thresholds (HCM 6th Edition).
• Signalized and unsignalized have different delay–LOS tables.
3. Volume to Capacity Ratio (v/c)
Shows congestion level and adequacy of lanes.
4. Queue Length

• Average queue
• 95th percentile queue
Provides evidence of spillback risk.
5. Approach Flow Rates
• Veh/hr for each lane group
• Movement distribution
6. Turning Movement Counts Validation
You already have 15-min volumes. These must align with simulation outputs.

B. Link/Segment Metrics
For corridors between intersections:

- Travel speed
- Density (veh/km/lane)
- Flow vs capacity
- Travel time through corridor
- Delay per segment

C. Operational Reliability
Useful but not always required:

- Demand vs supply imbalance
- Queue spillback blocking upstream intersections
- Left-turn storage adequacy
- Saturation flow rate checks

SUMO does not have these functions. SUMO is a microsimulation engine, not an HCM software. There is a lot of work to do in getting HCM performance metrics.

SUMO built-in outputs you can get:
1. Delay per edge or per lane
But this is **travel delay**, not **control delay** (HCM definition differs).
2. Queue Length
Using laneAreaDetectors or queue output.
You get:
• max queue
• mean queue
• queue length time series
3. Travel Time
You can specify key routes using <routeProbe>.
4. Density, flow, speed
Using edgeData or laneData.
5. Vehicle trajectories
For custom post-processing.

Computer SUMO Metrics to HCM Level outputs

Python module tia_tools.py
This module assumes you will

• Export edgeData or laneData for delay and v by c
• Export queueOutput or laneAreaDetector output for queues
• Use a simple mapping of intersection approaches to SUMO edges

Per edge or per approach you already have the pieces for:

1. Control delay per vehicle

From compute_edge_control_delay and compute_approach_metrics

- Average control delay in seconds per vehicle for an edge or approach
- Can be reported by analysis period and by movement group

2. Level of service (LOS)

From los_signalized, los_unsignalized, via compute_approach_metrics

- LOS letter for each approach based on HCM style delay thresholds
- You can tag approaches as signalized or unsignalized and report LOS accordingly

3. Volume to capacity ratio v divided by c

From compute_edge_v_by_c and compute_approach_metrics

- Flow in vehicles per hour from SUMO edge data
- Capacity in vehicles per hour from your EdgeConfig inputs
- v divided by c ratio for each approach or lane group

4. Ninety fifth percentile queue length

From compute_edge_queue_p95

- Queue length distribution per edge from queue output
- Ninety fifth percentile queue length (meters) per edge or per approach
- Combined with storage length you can flag storage deficiency

5. Basic flow, speed, density from raw edge data

Although not wrapped in a special function, the edgeData parser gives you:

- Period flow in vehicles per hour
- Mean speed in meters per second
- Density if you configure SUMO to output it

You can easily expose these as metrics in your reports.

6. Simple warrant related outputs

From the warrant functions and your fifteen minute or hourly volumes:

- Whether Eight hour volume warrant condition is satisfied
- Whether Four hour volume warrant condition is satisfied
- Whether Peak hour warrant condition is satisfied
- Hour by hour volumes on major and minor approaches that triggered the warrant decision

Run tia_metrics file to compute performance measures
Run usage_patternoutput_export to get output results
Run

You want the warrant engine tied explicitly to the 11th Edition MUTCD, not generic placeholders. Below is a module that:

- Implements **Warrant 1 (Eight Hour Vehicular Volume)** directly from **Table 4C-1** (Conditions A and B, with 100, 80, 70, 56 percent columns)

- Provides a **framework** for **Warrant 2 (Four Hour)** and **Warrant 3 (Peak Hour)** that expects you to supply digitized volume curves from Figures 4C-1/4C-2 and 4C-3/4C-4.

I am not hard-coding the 4C-1/4C-2/4C-3/4C-4 curves; you can plug in CSVs with points taken from those figures.

Use mutcd_1 and mutcd_2 python files

---

8. Present Results

Interpret the result and generate report. Need to work on the UI