1. **System Requirements:**

Real time simulation that sensitive to

1. Demand: # of vehicles on the road, the new demand in future time horizon
2. Supply: road outflow or inflow Capacity and freeway travel times
3. Path/ramp choice: route choice behavior or the percentage of turning traffic on ramp and offramp
4. **Assumption:**

Path is given and network is given

Kjam[l]

1. **Data structure**

L: Maxim number of links

V: maximum number of vehicles

K: alternative paths: K=0 primal path

I: link index

**[I] Vehicle data:**

Veh\_current\_link\_index[v] : Current link index, = 0: no start, 100 complete

Veh\_link\_matrix[v][i]: link id. of vehicle’s ith link along the its path: no link = -1: dynamic destination choice

Veh\_TA[v][i]: arrival time of vehicle v’s current paths to the ith link: entrance time

Veh\_TD[v][i] departure time of vehicle v’s current paths from ith link: exit time

Veh\_TA[v][i=0] is given as the arrival time of vehicle v

**[ii] ] Link entrance queue at time t**

LinkOutFlowCapacity[l][t] // external given by road conditions or weather conditions, incidents

LinkInFlowCapacity[l][t]

LinkTravelTime[l][t]: minimum link travel time )( default value: free flow, can be changed under weather conditions, speed limit change)

LinkCumulativeArrival[l][t] =0;

LinkCumulativeDeparture[l][t] =0;

Network Format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Link id | From node | To node | Capacity | Free-flow travel time |
|  |  |  |  |  |
|  |  |  |  |  |

Patha data format: Veh\_link\_matrix[v][i]

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| v | Origin | destination | I=0 | I=2 | I=3 |  |  |  |  |  |  |
| V0 |  |  | 1 | 2 | 3 | 4 |  |  |  |  |  |
| V1 |  |  |  |  |  |  |  |  |  |  |  |
| V2 |  |  |  |  |  |  |  |  |  |  |  |
| V |  |  |  |  |  |  |  |  |  |  |  |

1. **Algorithm for simulation**

**Step 1: prepare input:**

Vehicle path generation: prepare Veh\_link\_matrix[v][i] from shortest path or path routing algorithm, initial departure time Veh\_TA[v][i=0]

Veh\_TD[v][i=0] = Veh\_TA[v][i=0]+FFTT()

For vehicles on the link already, we need to determine the position of vehicle and calculate Veh\_TD[v][i=0]

Set LinkCumulativeArrival[l][t=0] and LinkCumulativeDeparture[l][t] for the vehicles already on the road

**Determine link capacity**

LinkOutFlowCapacity[l][t]

**Step 2: perform simulation**

for (simulation time t = 0; t <T; t++) //1st loop for each simulation time

{

// step 2.2. determine link capacity associated with bottleneck

For (link l =0; l <L; l++) //2nd loop for each link in the network

{

LinkNumberOfVehicles = LinkCumulativeArrival[l][t-1] - LinkCumulativeArrival[l][t-1]

LinkInFlowCapacity[n][t] = Kjam[l]\* linklength[l] - LinkNumberOfVehicles;

}

For (vehicle v = 0 ; v<V; v++) //2ndrd loop for each vehicle in the network

{

If needed, perform shortest path: if t== departure time of vehicle, or if real time information is available , we only need to replace the element in the Veh\_link\_matrix[v][i] arrary

Int i = Veh\_current\_link\_index[v]; // step 3.1: find the link sequence no. of vehicle v’ path

Int current\_link id\_c = Veh\_link\_matrix[v][i] // step 3.2: find current link id

If( vehicle v is on link i’s exit queue) i.e. Veh\_TD[v][i] == t

{ //step 3.3: tentative departure time = current time: ready to move

Int n = Veh\_link\_matrix[v][i+1] // next link sequence no of vehicle

If(LinkOutFlowCapacity[l][t] >=1 && LinkInFlowCapacity[n][t] >=1) // out and in capacity available

{ //step 3.4. check capacity

// step 3.5 : move

LinkOutFlowCapacity[l][t] -= 1; //reduce capacity

LinkInFlowCapacity [n][t] -= 1; //reduce capacity on next link

Veh\_current\_link\_index[v] +=1; // move to next link

//if the vehicle completes its trip: out of loop

// Veh\_TA[v][i+1] = t // arrives at the next link

Veh\_TD[v][i+1] = t+LinkTravelTime(j,t) // consider freeflow travel

} else

{//step 3.6: no move wait for one more time interval

Veh\_TD[v][i]= t+1; // wait for one more time interval

}

}

}//end for loop for each vehicle

} //end for loop for each time t

**Output**

Vehicle travel time, link arrival and departure time curves for each link -> flow, speed and density

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| V1: TA |  |  |  |  |  |
| V1: TD |  |  |  |  |  |
|  |  |  |  |  |  |

1. **Remarks and Comments:**

**Veh\_link\_matrix[v][i]:** is generated by the shortest path algorithm

Memory: 4 bytes for 32 bit integer; 1 vehicle-> 250 links: -> 1KB; 1MB -> 1000 vehicles; 1M vehicles -> 1GB

**Major innovative points:**

1. Real time mesoscopic traffic simulation with various time steps, suitable for any network: this is a simplified and improved version of DTALite
2. We can perform multiple scenarios based analysis, for stochastic weather and incident conditions. The algorithm can output important path travel time and link MOE (density, flow) for further analysis
3. This is a path based traffic simulation so we can evaluate the impact of real time routing strategies and different rerouting methods
4. **How to prepare data: step-by-step procedure**

Please contact Xuesong and Tony for latest NEXTA/DTALite version

Check [Data Structure and workflow of DTALite and NeXTA](https://docs.google.com/document/d/1z4YsztPXcWfQAd8NVD4_KXv7hJcbyXdYPzwL6z8xn3U)

Use NEXTA / DTALite

Step 1: Prepare input\_node.csv, input\_link.csv, input\_activity\_location.csv, input\_zone.csv

Step 2: Prepare input\_agent data base, and demand file (selected one of format, path data is not required at this stage)

Step 3: Run initial simulation run to check feasibility of network and demand: check output\_agent.csv and output\_summary, and output\_link\_MOE.csv

Step 4: According to different scenarios, refine network representation, jam density, link length, speed limit,

Step 5: According to different scenarios, refine demand representatino, different OD zones, different users/pricing/information classes, different link pricing strategies, and different work zone/capacity reduction strategies

Step 6: Perform refined simulation, generate reports (presentation due Dec 4th, 5-page report due Dec 15th)