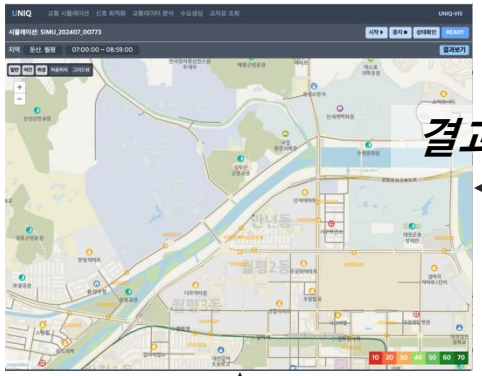
UNIQ-SALT 실행 가이드

개요

❍SALT 실행

- 입력데이터
- 결과데이터
- SALT 실행 방법
  - 도커 기반 실행 방법

UNIQ-SALT 실행



결과 가시화

실시간 가시화

```
{
  "scenario": {
    "id": "sim_doan",
    "time": {
      "begin": 25200,
      "end": 32280
    },
    "input": {
      "fileType": "SALT",
      "node": "node.xml",
      "link": "edge.xml",
      "connection": "connection.xml",
      "trafficLightSystem": "tss.xml",
      "multiarea": "multiarea/edge.xml",
      "route": "doan.rou.xml"
    },
    "parameter": {
      "minCellLength": 30,
      "vehLength": 5
    },
    "output": {
      "fileDir": "/uniq/simulator/data/sim_doan/output",
      "period": 600,
      "level": "cell",
      "save": 1
    }
  }
}
```

JSON 파일

시나리오 파일

UNIQ-SALT 시뮬레이터

결과 파일

CSV 파일

XML 파일

node

edge

connection

tss

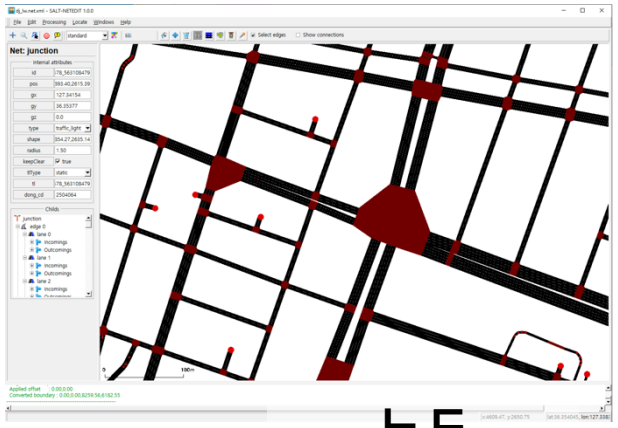
route

시나리오 파일

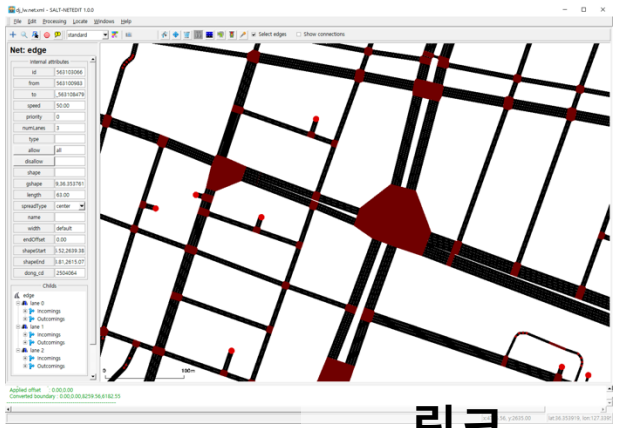
- **id**
  - 시뮬레이션 ID
  - 해당 ID 값을 결과 파일명에 사용
- **time**
  - 시뮬레이션 시작 (begin) 과 종료 (end) 시간을 명시
  - (현재) 24시간 초단위로 사용 중, 예) 25200 → 오전 7시 의미
- **input**
  - 입력데이터의 상대경로 포함 URL 정보
- **parameter**
  - 시뮬레이션 설정값
  - (현재) 셀길이, 차량길이 정보만
    - 원래 의도는 차량 행동 관련 설정값 정보를 넘기기 위한 attribute
- **output**
  - 결과파일 생성과 관련된 설정값
  - fileDir: 결과파일 생성 위치 URL 지정
  - period: 결과 데이터 통계 산출 시간구간, 예) 600 → 10분 단위로 결과 통계 산출/저장
  - level: 셀 단위 (cell), 링크 단위 (link) 산출
  - save: 저장 (1)

```
{
  "scenario": {
    "id": "sim_doan",
    "time": {
      "begin": 25200,
      "end": 32280
    },
    "input": {
      "fileType": "SALT",
      "node": "node.xml",
      "link": "edge.xml",
      "connection": "connection.xml",
      "trafficLightSystem": "tss.xml",
      "multiarea": "multiarea/edge.xml",
      "route": "doan.rou.xml"
    },
    "parameter": {
      "minCellLength": 30,
      "vehLength": 5
    },
    "output": {
      "fileDir": "/uniq/simulator/data/sim_doan/output",
      "period": 600,
      "level": "cell",
      "save": 1
    }
  }
}
```

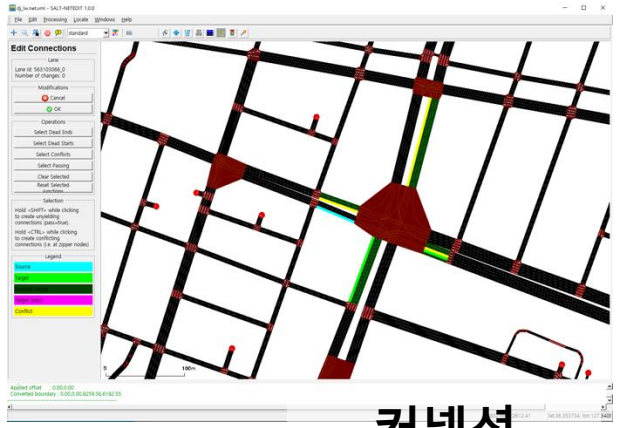
입력데이터: 노드, 링크, 커넥션



노드



링크



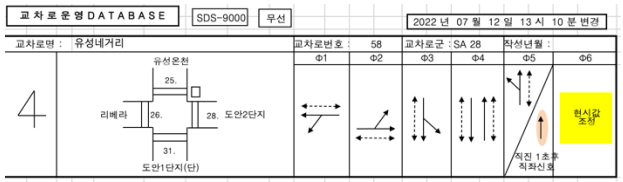
커넥션

```
<nodes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/nodes_file.xsd">
    <node id="553800278" x="127.32328" y="36.33627" z="0.0" type="priority" radius="1.5" keepClear="true" dong_cd="2504064"/>


<edges xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/edges_file.xsd">
    <edge id="-553800929" from="553801010" to="553801008" priority="0" numLanes="1" speed="10.0" shape="127.334784,36.327522
127.33479,36.327321" dong_cd="2504051" edge_len="22" left_pocket="-1" right_pocket="-1" blanest="0"/>


<connections xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/connections_file.xsd">
    <connection from="553800929" to="553804241" fromLane="0" toLane="0"/>
    <connection from="553800929" to="553804260" fromLane="0" toLane="0"/>
    <connection from="553800929" to="553810515" fromLane="0" toLane="0"/>
```

입력데이터: 신호

교차로운영DATABASE | SDS-9000 | 무선 | 2022 년 07 월 12 일 13 시 10 분 변경

교차로명 : 유성네거리 | 교차로번호 : 58 | 교차로군 : SA 28 | 작성년월 :



교차로 구분		A1	A2	A3	A4	A5	A6
교차로 운영	RING A						
	RING B						
		B1	B2	B3	B4	B5	B6

SIGNAL TABLE

OPTION	10	20	10	10	10	
	10	20	10	10	10	
MIN. GREEN	39	42	32	20	14	
	39	42	32	20	14	
MAX. II	70	75	55	35	40	
	70	75	55	35	40	
YELLOW(RED)	3	3	3	3	3	
	3	3	3	3	3	
BEF. PED.		2		2		
	2		2			
WALK		16		12		
	14		13			
WALK CLEAR		24		9	14	
	23		20	3		

PLAN TABLE

TIME PLAN 1

수준	주기(초)	ID	연동	시간분할(초)	감응제어 RING A, B	압버튼(×)	검지기(○)	검지기(×)
1	180	1	52	47,49,40,23,21				
		2	50	45,45,35,23,32				
2	180			45,45,35,23,32				
		3	40	47,49,40,23,21				
		4	40	47,49,40,23,21				
		5	52	47,49,40,23,21				
3	180			47,49,40,23,21				
		6	52	47,49,40,23,21				
		7	52	47,49,40,23,21				
4		8						
		9						
		10						
5		11						
		12						
		13						
6		14						
		15						
		16						

구분	TOD PLAN 1 시각	수준	ID	TOD PLAN 2 시각	수준	ID	TOD PLAN 3 시각	수준	ID	시작	종료	요일/휴일
1	00:00	1	1	00:00	1	1	00:00	1	1			
2	07:00	2	2	07:00	2	2	09:30	2	3			
3	09:30	2	3	09:30	2	3	21:00	3	6			
4	17:00	3	5	11:30	3	5						
5	21:00	3	6	21:00	3	6						
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												

예약정보(월~일 : 0, 토:7,일:1,월~금:8)

주간신호계획

요일	TOD PLAN	TIME PLAN
일	3	1
월	1	1
화	1	1
수	1	1
목	1	1
금	1	1
토	2	1

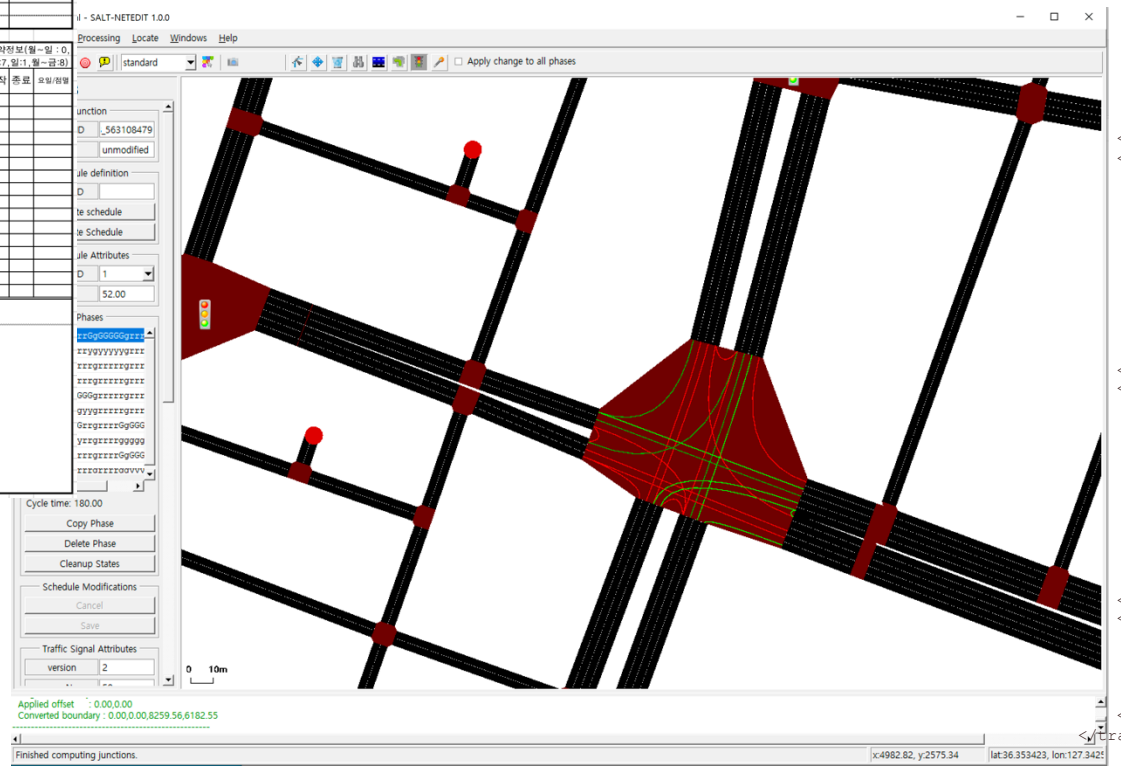
특수일신호계획

요일	TOD PLAN	TIME PLAN
1.1	3	1
3.1	3	1
5.5	3	1
6.6	3	1
8.15	3	1
10.3	3	1
12.25	3	1

MEMO

LSU사용 방법

LSU사용은 북동남서순으로…
그외에는 LSU5번부터 사용…
차량보조등은 LSU8번을 사용

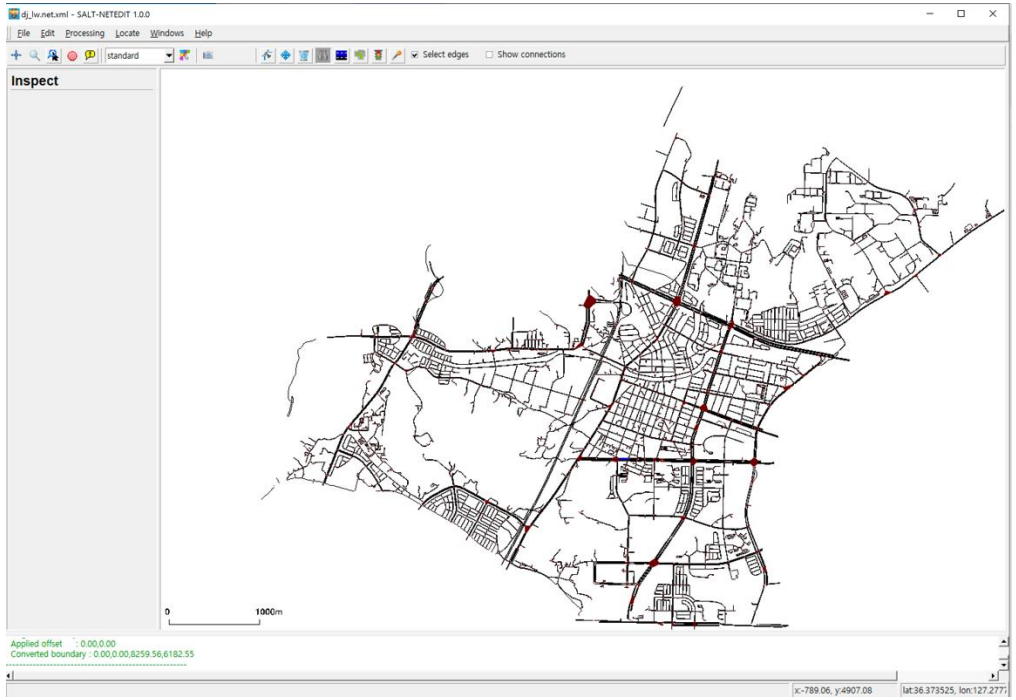


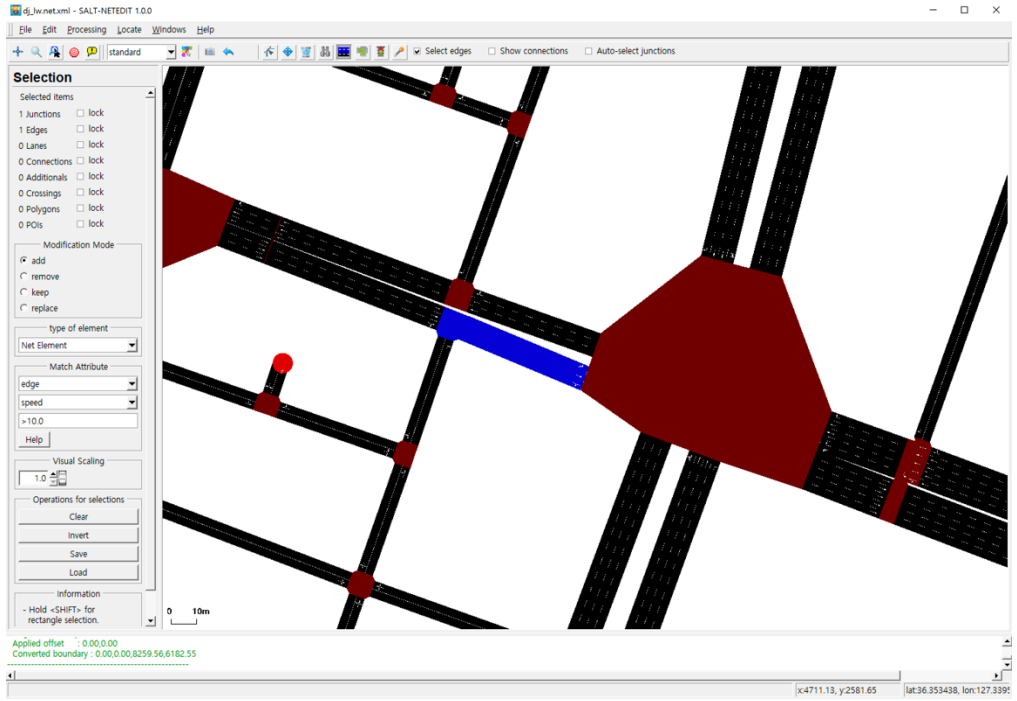

```
<trafficSignal
nodeID="cluster_563101520_563101521_563101678_563101679_563102728_563102729_563102732_563102733_563102853_563103948_563103950_563103951_563108472_563108475_563108478_563108479" version="2"
crossNo="58" crossName="1번 (유성네거리)" date="2022-07-12" signalGroup="28">
    <schedule offset="52" id="1">
        <phase state="grrrrrGgGGGGGgrrrrrrgrrrrr" duration="44" minDur="39" maxDur="70"/>
        <phase state="grrrrrygyyyyygrrrrrryrrrrr" duration="3"/>
        <phase state="grrrrrrgrrrrrgrrrrrGrGGGGG" duration="46" minDur="42" maxDur="75"/>
        <phase state="grrrrrrgrrrrrgrrrrryryyyyg" duration="3"/>
        <phase state="gGGGGGGgrrrrrgrrrrrrrrrrrG" duration="37" minDur="32" maxDur="55"/>
        <phase state="gggggyyygrrrrrgrrrrrrrrrrg" duration="3"/>
        <phase state="gGGGGrrgrrrrrGgGGGGrrrrrrrG" duration="20" minDur="20" maxDur="35"/>
        <phase state="gyyyyrrgrrrrrggggggrrrrrrry" duration="3"/>
        <phase state="grrrrrrgrrrrGgGGGGGGgrrrrr" duration="18" minDur="14" maxDur="40"/>
        <phase state="grrrrrrgrrrrrgggyyyyyygrrrrr" duration="3"/>
    </schedule>
    <schedule offset="50" id="2">
        <phase state="grrrrrGgGGGGGgrrrrrrgrrrrr" duration="42" minDur="39" maxDur="70"/>
        <phase state="grrrrrygyyyyygrrrrrryrrrrr" duration="3"/>
        <phase state="grrrrrrgrrrrrgrrrrrGrGGGGG" duration="42" minDur="42" maxDur="75"/>
        <phase state="grrrrrrgrrrrrgrrrrryryyyyg" duration="3"/>
        <phase state="gGGGGGGgrrrrrgrrrrrrrrrrrG" duration="32" minDur="32" maxDur="55"/>
        <phase state="gggggyyygrrrrrgrrrrrrrrrrg" duration="3"/>
        <phase state="gGGGGrrgrrrrrGgGGGGrrrrrrrG" duration="20" minDur="20" maxDur="35"/>
        <phase state="gyyyyrrgrrrrrggggggrrrrrrry" duration="3"/>
        <phase state="grrrrrrgrrrrGgGGGGGGgrrrrr" duration="29" minDur="14" maxDur="40"/>
        <phase state="grrrrrrgrrrrrgggyyyyyygrrrrr" duration="3"/>
    </schedule>
    <schedule offset="40" id="3">
        <phase state="grrrrrGgGGGGGgrrrrrrgrrrrr" duration="44" minDur="39" maxDur="70"/>
        <phase state="grrrrrygyyyyygrrrrrryrrrrr" duration="3"/>
        <phase state="grrrrrrgrrrrrgrrrrrGrGGGGG" duration="46" minDur="42" maxDur="75"/>
        <phase state="grrrrrrgrrrrrgrrrrryryyyyg" duration="3"/>
        <phase state="gGGGGGGgrrrrrgrrrrrrrrrrrG" duration="37" minDur="32" maxDur="55"/>
        <phase state="gggggyyygrrrrrgrrrrrrrrrrg" duration="3"/>
        <phase state="gGGGGrrgrrrrrGgGGGGrrrrrrrG" duration="20" minDur="20" maxDur="35"/>
        <phase state="gyyyyrrgrrrrrggggggrrrrrrry" duration="3"/>
        <phase state="grrrrrrgrrrrGgGGGGGGgrrrrr" duration="18" minDur="14" maxDur="40"/>
        <phase state="grrrrrrgrrrrrgggyyyyyygrrrrr" duration="3"/>
    </schedule>
    <schedule offset="52" id="5">
        <phase state="grrrrrGgGGGGGgrrrrrrgrrrrr" duration="44" minDur="39" maxDur="70"/>
        <phase state="grrrrrygyyyyygrrrrrryrrrrr" duration="3"/>
        <phase state="grrrrrrgrrrrrgrrrrrGrGGGGG" duration="46" minDur="42" maxDur="75"/>
        <phase state="grrrrrrgrrrrrgrrrrryryyyyg" duration="3"/>
        <phase state="gGGGGGGgrrrrrgrrrrrrrrrrrG" duration="37" minDur="32" maxDur="55"/>
        <phase state="gggggyyygrrrrrgrrrrrrrrrrg" duration="3"/>
        <phase state="gGGGGrrgrrrrrGgGGGGrrrrrrrG" duration="20" minDur="20" maxDur="35"/>
        <phase state="gyyyyrrgrrrrrggggggrrrrrrry" duration="3"/>
        <phase state="grrrrrrgrrrrGgGGGGGGgrrrrr" duration="18" minDur="14" maxDur="40"/>
        <phase state="grrrrrrgrrrrrgggyyyyyygrrrrr" duration="3"/>
    </schedule>
    <schedule offset="52" id="6">
        <phase state="grrrrrGgGGGGGgrrrrrrgrrrrr" duration="44" minDur="39" maxDur="70"/>
        <phase state="grrrrrygyyyyygrrrrrryrrrrr" duration="3"/>
        <phase state="grrrrrrgrrrrrgrrrrrGrGGGGG" duration="46" minDur="42" maxDur="75"/>
        <phase state="grrrrrrgrrrrrgrrrrryryyyyg" duration="3"/>
        <phase state="gGGGGGGgrrrrrgrrrrrrrrrrrG" duration="37" minDur="32" maxDur="55"/>
        <phase state="gggggyyygrrrrrgrrrrrrrrrrg" duration="3"/>
        <phase state="gGGGGrrgrrrrrGgGGGGrrrrrrrG" duration="20" minDur="20" maxDur="35"/>
        <phase state="gyyyyrrgrrrrrggggggrrrrrrry" duration="3"/>
        <phase state="grrrrrrgrrrrGgGGGGGGgrrrrr" duration="18" minDur="14" maxDur="40"/>
        <phase state="grrrrrrgrrrrrgggyyyyyygrrrrr" duration="3"/>
    </schedule>
    <TODPlan offset="52" defaultPlan="1">
        <plan offset="50" schedule="2" startTime="25200"/>
        <plan offset="40" schedule="3" startTime="34200"/>
        <plan offset="52" schedule="5" startTime="61200"/>
        <plan offset="52" schedule="6" startTime="75600"/>
    </TODPlan>
</trafficSignal>
```

[참고] SALT NetEdit

❍SALT용 도로망/신호 데이터 편집도구

- SHP to SALT XML 변환
- SALT XML 기반 도로망/신호 가시화
- 노드/링크/커넥션 편집 가능
- 신호 편집 가능 → XML 로만 저장





입력데이터: 수요 (차량별 궤적)

```
<routes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/routes_file.xsd">
    <vehicle id="0" depart="25200.0" departLane="best" departPos="random" departSpeed="max">
        <route edges="-563104010 563103690 -563103691 563101761 563103264 563108181 563108179 -563111304 563101159
563101251 563108183" />
```

- **시뮬레이션 중에 투입될 모든 차량의 궤적 정보 포함**
- **(현재) Static Demand 로 제공**
  - Static Demand → 시뮬레이션 상황에 관계없이 경로는 고정
  - 차량 혼잡 상황 반복 재현 가능
  - (한계 1) 교통 역량 평가의 경우, 교통 수요 변화 필수 → Dynamic Demand 제공 필요
    - Dynamic Demand → 시뮬레이션 도로 통행 상황을 고려하는 경로
      - ✓ (일반적으로) 특정 주기마다 라우팅 테이블을 재산출하여 User Equilibrium을 맞추도록 반복 할당 수행을 통해 최종 할당 경로 데이터 산출 확보, 예) SUMO DUA Routing
    - Dynamic Demand 산출 후 최종 궤적 데이터는 Static 경로로 제공
  - (한계 2) 돌발 상황에서의 에이전트 행동 특성에 따른 경로 변화 상황 반영이 어려움
    - 상황 재현이 아니라 예측에서 필요한 기능
    - 돌방 상황 발생에 영향 받는 범위 내 차량 에이전트에 대한 동적 경로 변경 기능 제공

결과데이터

```
intervalbegin,intervalend,roadID,VehPassed,AverageSpeed,AverageDensity,WaitingQLength,WaitingTime,SumTravelLength,SumTravelTime
```

항목	의미	비고
intervalbegin	통계구간 시작 타임스텝, 예) 5분단위 통계 → 0, 300, 600, …	
intervalend	통계구간 종료 타입스텝, 예) 5분단위 통계 → 299, 599, 899, …	
roadID	결과 산출 셀 ID 혹은 링크 ID	시나리오 파일의 Output level 값에 따라 정해짐
VehPassed	통계구간 동안 총 통과차량 수	
AverageSpeed	통계구간 동안 평균 통과속도	
AverageDensity	통계구간 동안 평균 밀도	
WaitingQLength	통계구간 동안 대기 큐 길이 평균 기대값	
WaitingTime	통계구간 동안 평균 대기 시간	
SumTravelLength	통계구간 동안 통과차량의 통과거리 합	
SumTravelTime	통계구간 동안 통과차량의 통과시간 합	

[참고] 동적 인터페이스 GET 함수 (셀 기준)

구분	항목	의미	비고
시간 구간 T 동안의 수집/산출	getNumBehPassed	신호 교차로 진입 링크 대상으로 시간 구간 T 동안 통과한 차량 수 (링크/레인/셀 별)	
	getAverageSpeed	신호 교차로 진입 링크 대상으로 시간 구간 T 동안 통과한 차량의 평균 링크/레인/셀 별 속도	
	getAverageDensity	신호 교차로 진입 링크 대상으로 시간 구간 T 동안 평균 링크/레인/셀 별 밀도	
	getAverageVehicleWaitingTime	신호 교차로 진입 링크 대상으로 시간 구간 T 동안 링크/레인/셀 별 차량들의 평균 대기 시간	
	getAverageVehicleWaitingQLength	신호 교차로 진입 링크 대상으로 시간 구간 T 동안 링크/레인/셀 별 대기열 길이 합	
	getSumTravelTime	신호 교차로 진입 링크 대상으로 시간 구간 T 동안 통과한 차량들의 링크/레인/셀 별 통과 시간 합	
특정 시간에서 수집	getCurrentAverageWaitingTimeBasedVehicle	특정 시간에, 신호 교차로 진입 링크 대상으로 링크/레인/셀 별 운행 중인 모든 차량들의 평균 대기 시간	
	getCurrentWaitingTimeSumBaseVehicle	특정 시간에, 신호 교차로 진입 링크 대상으로 링크/레인/셀 별 운행 중인 모든 차량들의 대기 시간 합	

시뮬레이션 실행

❍가시화 서버를 이용

- http://129.254.177.53:8081
- ETRI 내부에서만 접속

❍GIT 소스 or 오픈 라이브러리 활용

- Private GIT
  - 소스 오픈 이슈
- Public GIT
  - SALT 정적 라이브러리 → 정적 라이브러리 but 라이브러리 생성 시스템 환경 (버전 등) 영향 존재

❍도커 활용

- 합성데이터용 SALT 도커 이미지를 생성 후 제공
- 도커 사용을 위한 id/passwd 공유 필요

Docker 사용을 위한 사전 환경 설정

❍docker 설치

- centos 기준
  - 참고: https://docs.docker.com/engine/install/centos/
- ubuntu 기준
  - 참고: https://jjeongil.tistory.com/1968

❍docker hub를 통한 이미지 공유

- (로그인) images4aixsim / (이미지:태그) salt:v2.1a.240903
  - $ sudo docker login images4aixsim
  - $ sudo docker pull images4aixsim/salt:v2.1a.240903

Docker 사용 UNIQ-SALT 실행

❍ 시나리오 파일을 특정 폴더에 위치

- 예) /home/xxx/uniq_sim/sim_doan
- sim_doan 디렉토리 아래
  - node.xml/edge.xml/connection.xml/tss.xml → 도로망 & 신호
  - doan.rou.xml → 수요
  - salt.scenario.json → 시나리오 파일
    - ✓ 해당 시나리오 파일 내 도로망&신호&수요 데이터 파일에 대한 URL 정보가 포함되어 있음

❍ 도커 명령어를 사용하여 시뮬레이션 실행 (기본 실행)

- docker run 이용
  - docker run –v [로컬 디렉토리 위치]:[도커 내 마운트 위치] [도커허브ID]/[이미지명] [실행 명령어]
  - (예) $ **docker run -v** /home/xxx/uniq_sim:/uniq/simulator/data images4aixsim/salt:v2.1a.240903 **python** bin/salt.py **-s** /uniq/simulator/data/sim_doan/salt.scenario.json
- 기본 실행 외, 원하는 시뮬레이션 실행을 위한 python 코드를 작성 후, 도커에 마운팅하는 볼륨에 해당 코드를 저장하여 실행 가능
  - (예) 실행용 python 코드(salt.py) 를 /home/xxx/uniq_sim/sim_python 에 저장한 경우,
  - $ **docker run -v** /home/xxx/uniq_sim:/uniq/simulator/data images4aixsim/salt:v2.1a.240903 **python** /uniq/simulator/data/sim_python/salt.py **-s** /uniq/simulator/data/sim_doan/salt.scenario.json

[참고] 시뮬레이션 실행을 위한 Python 코드 작성 예시

```
import argparse
import math
import os
import json
import re
import libsalt

#cwd_path = os.getcwd()
#this_file_dir_path = os.path.dirname(os.path.realpath(__file__))
#salt_core_root_dir = os.path.join(this_file_dir_path, "../..")
#default_scenario_path = os.path.join(salt_core_root_dir, "data", "scenario.default.json")

if 'SALT_HOME' in os.environ:
    salt_home_dir =os.environ['SALT_HOME']
else:
    sys.exit("Please declare the environment variable 'SALT_HOME'")

default_scenario_path = os.path.join(salt_home_dir, "data", "scenario.default.json")

def parse_args():
    parser = argparse.ArgumentParser(description="Run Dynamic Simulation")
    parser.add_argument('-s', '--scenario', nargs='?', default=default_scenario_path)

    return parser.parse_args()

def getJsonObj(_jsonPath):
    f = open(_jsonPath, "r")
    #return json.load(f)
    data = re.sub("//.*", "", f.read(), re.MULTILINE)
    return json.loads(re.sub("//.*", "", data, re.MULTILINE))

def main():
    args = parse_args()
    #scenarioPath = os.path.join(cwd_path, args.scenario)
    scenarioPath = args.scenario
    print('args', args)
    print('scenario path', scenarioPath)

    scenarioJson = getJsonObj(scenarioPath)
    beginStep = scenarioJson['scenario']['time']['begin']
    endStep = scenarioJson['scenario']['time']['end']


    libsalt.start(scenarioPath)
    libsalt.setCurrentStep(beginStep)
    step = libsalt.getCurrentStep()
    while step <= endStep:
        libsalt.simulationStep()
        step = libsalt.getCurrentStep()

    libsalt.close()
    print("Python: Simulation End!!!")

    exit(0)

main()
```

- 시나리오 파일 파싱
- 시뮬레이션 수행 시작
- 시뮬레이션 수행 메인 루프
- 시뮬레이션 수행 종료