

신호 최적화와 시각화 연동

2022/04/14

mgpi 작성 신호 최적화 설명 자료(2022/03/29)에 기반하여 수정

도커 이미지 최적화 등록

<https://hub.docker.com/r/images4uniq/optimizer/tags>

The screenshot shows the Docker Hub page for the repository `images4uniq/optimizer`. The page includes a search bar, navigation links (Explore, Pricing, Sign In, Register), and a list of tags. The tags list shows several versions, with the most recent being `v0.1a.20220404`. A blue arrow points from the text `docker pull images4uniq/optimizer:v0.1a.20220404` to the tag `v0.1a.20220404` in the list.

docker pull [images4uniq/optimizer:v0.1a.20220404](#)

The screenshot shows the UNIQ application interface. The main window displays a table of experiments with columns for EXP ID, 저장주기 (Save Cycle), Epoch, and Duration. A modal window titled "최적화 등록" (Optimization Registration) is open, showing fields for simulation ID, description, region, and simulation start/end times. A blue arrow points from the text `images4uniq/optimizer:v0.1a.20220404` to the "실행 이미지" (Execution Image) field in the modal.

UNIQ 시뮬레이션 최적화환경 실행최적화 실행환경 모니터링

+ 등록

| EXP ID | 저장주기 | Epoch | Duration |
|------------------|------|-------|----------|
| EXP_202110_00596 | 20 | 100 | 7140 |
| EXP_202110_00160 | 20 | 10 | 714 |
| EXP_202111_00328 | 20 | 10 | 7140 |
| EXP_202111_00235 | 20 | 10 | 714 |

교차로: SA101 SA107 SA111

실행 이미지: images4uniq/optimizer:v0.1a.20220404

도커 이미지 사용 실행 : 입력(시나리오) 준비

1. 특정 경로 밑에 아래와 같이 scenario 폴더 구성

```
(base) pi@pi-System-Product-Name:~$ tree new_docker_test/
new_docker_test/
├── scenario
│   ├── dj_all
│   │   ├── dj_all.connection.xml
│   │   ├── dj_all.edge.xml
│   │   ├── dj_all.node.xml
│   │   ├── dj_all.rou.xml
│   │   ├── dj_all_simulate.scenario.json
│   │   ├── dj_all_test.scenario.json
│   │   ├── dj_all_train.scenario.json
│   │   └── dj_all.tss.xml
│   ├── doan
│   │   ├── doan.connection.xml
│   │   ├── doan.edge.xml
│   │   ├── doan.node.xml
│   │   ├── doan.rou.xml
│   │   ├── doan_simulate.scenario.json
│   │   ├── doan_test.scenario.json
│   │   ├── doan_train.scenario.json
│   │   └── doan.tss.xml
│   └── sa_1_6_17
│       ├── sa_1_6_17.connection.xml
│       ├── sa_1_6_17.edge.xml
│       ├── sa_1_6_17.node.xml
│       ├── sa_1_6_17.rou.xml
│       ├── sa_1_6_17_simulate.scenario.json
│       ├── sa_1_6_17_test.scenario.json
│       ├── sa_1_6_17_train.scenario.json
│       └── sa_1_6_17.tss.xml
4 directories, 24 files
```

dj_all - 대전 전체 수요 데이터 및 전체 시간 수요 데이터

doan - 2차년도 도안 지역 7-9시 실험을 위한 데이터

sa_1_6_17 - 전체 수요 데이터로 7-9시, 7-12시 수요 실험을 위한 데이터

- 현재는 시나리오 이름으로 위 3개만 허용
 - 다른 폴더명 인식 불가
 - 안에 세부 파일 내용은 변경 가능

도커 이미지 사용 실행 : 입력(시나리오) 준비

참고 : 시나리오 추가시 폴더 내 파일 구성

시나리오 추가시 경로에는 SALT 실행에 필요한 파일이 모두 있어야 하며

아래와 같이 이름들을 맞춰야 함

- ex. dunsan 환경 추가시,

1. data/envs/salt/ 아래에 **dunsan** 폴더 추가 : 폴더 명은 map 이름으로 사용됨

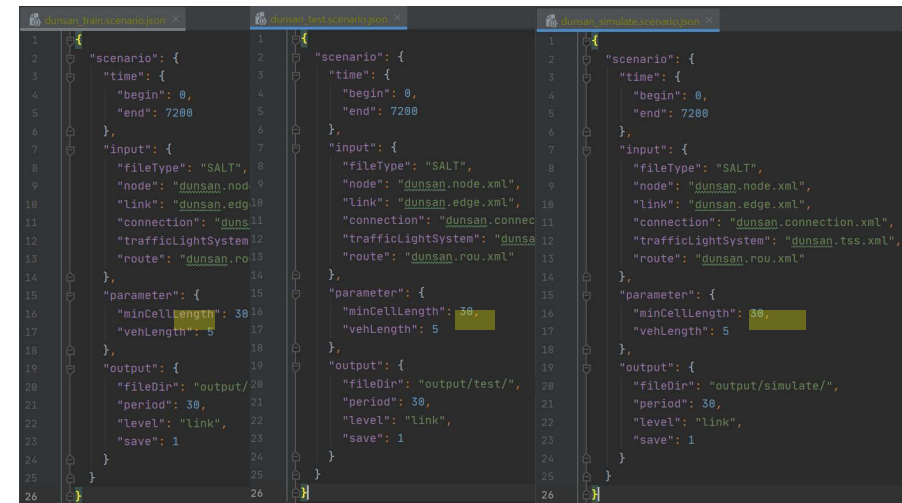
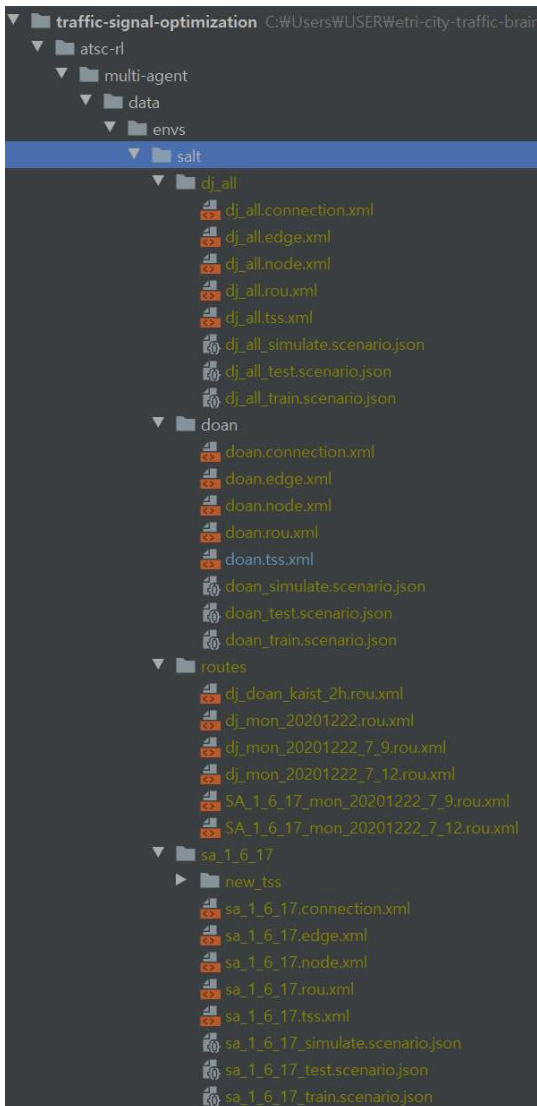
2. **dunsan** 폴더 아래에 아래 파일들 추가

- **dunsan.connection.xml**
- **dunsan.edge.xml**
- **dunsan.node.xml**
- **dunsan.tss.xml**
- **dunsan.rou.xml**
- **dunsan_train.scenario.json**
- **dunsan_test.scenario.json**
- **dunsan_simulate.scenario.json**

mode 별 별도 파일 {args.map}_{args.mode}.scenario.json

3. run.py에서 --map argument에 폴더 이름 추가

```
parser.add_argument('--map', choices=['dj_all', 'doan', 'sa_1_6_17'], default='sa_1_6_17')  
parser.add_argument('--map', choices=['dj_all', 'doan', 'sa_1_6_17', 'dunsan'], default='sa_1_6_17')
```



도커 이미지 사용 실행 : docker run...

빨간 글씨는 고정하면 좋을 듯.

2. `sudo docker run -v {scenario폴더 상위 경로}:/uniq/optimizer/io {실행할 이미지} python run.py {args} -
-io-home io --scenario-file-path io/scenario`

공유할 디렉토리를

(반드시) /uniq/optimizer의 하부 디렉토리로 마운트

실행할 이미지

```
(base) pi@pi-System-Product-Name:~$ sudo docker run -v /home/pi/new_docker_test:/uniq/optimizer/io images4uniq/optimizer:v0.1a.20220404 python run.py --mode train  
--map 'doan' --action offset --method sappo --target-TL "SA 101" --epoch 1 --io-home io --scenario-file-path io/scenario --result-comp "FALSE"
```

```
[%] sudo docker run -v /home/tsoexp/z.docker_test/io:/uniq/optimizer/io images4uniq/optimizer:v0.1a.20220404  
python run.py --mode train --map doan --action offset --method sappo --target-TL 'SA 101'  
--start-time 0 --end-time 7200 --epoch 1 --io-home io --scenario-file-path io/scenario  
--result-comp false
```

3. train mode 실행하고 나면 logs, model, output 경로에 파일들이 생성 됨

- logs/ - 가시화 서버에선 사용 x
- model/ - test mode 실행시 필요한 모델 생성
- output/ - 가시화를 위한 파일들 생성(보상 그래프, phase 그래프)

실행 인자가 변경되었으므로
인자 받아들이는 UI
명령어 생성 로직 수정 필요할 듯

도커 이미지 사용 실행 : docker run...

참고 : 최적화 주요 실행 인자

| 인자명 | 설 명 |
|--------------------|---|
| mode | 훈련(train)과 추론(test) 여부를 나타내는 'train', 'test', 고정 시간 시나리오인 'simulate' 중 하나를 가짐. default 는 'train' |
| model-num | 추론시 사용할 저장된 모델 번호로 default는 0 |
| result-comp | default는 "TRUE", 가시화서버에서는 "FALSE"로 사용 |
| start-time | 시뮬레이션 시작 시간(스텝) : 여기 설정 값과 시나리오 파일의 begin 값 중 큰 값이 시작 시간 |
| end-time | 시뮬레이션 종료 시간 : 여기 설정 값과 시나리오 파일의 end 값 중 작은 값이 종료 시간 |
| epoch | 훈련(학습) 반복 회수로 default는 3000 |
| model-save-period | 훈련 중 모델 저장 주기로, default는 20 |
| target-TL | 신호 최적화 대상으로 default는 "SA 101,SA 104,SA 107,SA 111"임. 신호 그룹을 콤마로 연결(예, --target-TL SA 101,SA 104) |
| reward-func | 강화학습의 보상 함수로 ['pn', 'wt', 'wt_max', 'wq', 'wt_SBV', 'wt_SBV_max', 'wt_ABV'] 중 하나, default는 'wt_SBV' pn은 통과 차량 수, wt는 대기시간, wq는 대기 큐 길이, SBV 는 sum에 기반한 보상값을, ABV는 avg에 기반한 보상값을 의미 |
| io-home | 최적화 중 생성되는 파일들이 저장되는 곳의 최상위 디렉토리로 최적화 기본 디렉토리인 /uniq/optimizer 로부터 상대 경로로 표시 |
| scenario-file-path | 사용할 시나리오 파일이 위치한 경로로, 기본 디렉토리인 /uniq/optimizer 로부터 상대 경로로 표시(io/scenario 고정) |
| map | 최적화 대상 지역('doan', 'dj_all', 'sa_1_6_17') |
| method | 최적화 모델 선택, default는 sappo ['sappo', 'ddqn', 'sappo_rnd'] |
| action | 최적화 action 선택, default는 offset ['kc', 'offset', 'gr', 'gro'] |

주의 : 사용 수요 파일(*.rou.xml)의 depart 확인하여 실행 인자(start/end) 설정

<vehicle id="0" depart="0.00" departLane="best" departPos="random" departSpeed="max">

Backup slides

- 신호 최적화 실행
 - `python run.py --mode train --map "sa_1_6_17" --target-TL "SA 6"`
--start-time 0 --end-time 7200
 - `python run.py --mode train --map "doan" --target-TL "SA 56"`
- 모두텍 변경

모두텍에 변경 요청해야 할 부분

[uniq/12.최적화/ppt/2022/코드 리뷰/도커 이미지를 이용한 신호 최적화 실행-20210923 mgpi 20220404수정.pptx](#)



The screenshot shows a file explorer window with a directory tree on the left and a table of files on the right. The directory tree includes folders like 'node_modules', 'public', 'routes', 'salt-connector', 'sim-runner', and 'test'. The table lists files with columns for filename, size, type, date, and status.

| 파일명 | 크기 | 파일 유형 | 최종 수정 | 권한 |
|----------------------|-------|---------------|-----------------------|-----------|
| download-files.js | 715 | JavaScript 파일 | 2021-11-04 오후 2:01:25 | -rw-rw... |
| exec-optimization.js | 3,082 | JavaScript 파일 | 2021-11-04 오후 2:01:25 | -rw-rw... |
| exec-remote.js | 1,930 | JavaScript 파일 | 2021-11-04 오후 2:01:25 | -rw-rw... |
| exec-simulation.js | 2,593 | JavaScript 파일 | 2021-11-04 오후 2:01:25 | -rw-rw... |
| exec.js | 684 | JavaScript 파일 | 2021-11-04 오후 2:01:25 | -rw-rw... |
| index.js | 128 | JavaScript 파일 | 2021-11-04 오후 2:01:25 | -rw-rw... |