Sappo 알고리즘 확장 수정(안)

(SappoEnv.py, SappoActionMgmt.py, SappoRewardMgmt.py)

- YJLEE 제안 내용 구현에 Sappo* 활용 가능성 검토
- YJLEE 검토 의견 반영을 위한 수정(안)
- Action 적용 관련 코드 설명

2023. 03.

YJ.Lee's idea 검토 : 교차로별 주기가 달라지면 SappoEnv에 적용 불가

동일 SA(Sub-Area, 교차로 그룹)에 속하는 교차로들의 주기는 동일하다.

Proposed I

❖ 63:24:37:26(주기 150)

ETRI

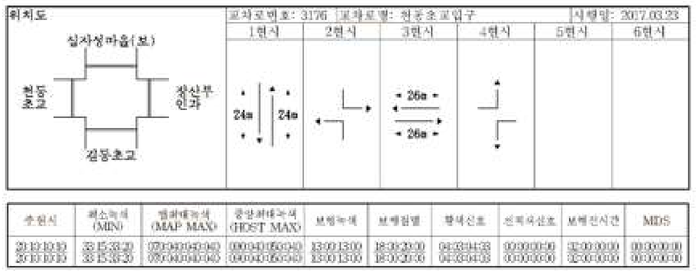
발췌 : Action Space 코드 수정 (v20230302, 이용진) 2p

	Phase 1	Phase 2	Phase 3	Phase 4	Total
Min	33.0	15.0	33.0	20.0	101.0
Median	51.5	27.5	36.5	30.0	145.5
Max	70.0	40.0	40.0	40.0	190.0
Action	$-1 \le a_1 \le +1$	$-1 \le a_2 \le +1$	$-1 \le a_3 \le +1$	$-1 \le a_4 \le +1$	$101 \le T \le 190$

$$T = (51.5 + a_1 \times 18.5) + (27.5 + a_2 \times 12.5) + (36.5 + a_3 \times 3.5) + (20.0 + a_4 \times 10.0)$$

- For each intersection, #Action outputs = #Phases
  - 신호 조합이 아닌, #Phases에 선형 비례로 증가.
- Action space를 4차원 vector로 표현
  - 모든 action 조합을 표현할 수 있음.
  - 비슷한 action vector는 실제로도 유사한 제어 신호임.
- 제약 조건
  - 최소/최대 녹색 시간 만족
  - 주기 (T=150)는 만족하지 않음.
    - Penalty 추가 (추후 고려)

천동초교입구



주현시	최소녹색 (MIN)	맵최대녹색 (MAP MAX)	중앙최대녹색 (HOST MAX)	보행녹색	보행점멸	황색신호	전적색신호	보행잔시간	MDS
[illegible]	33 15 33 20	070 040 040 040	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	33 15 33 20	070 040 040 040	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

SAPPO는 동일 SA(교차로 그룹)에 속하는 모든 교차로의 주기가 변경되지 않음을 가정하고 있기에 적용 불가

새로운 Env, ActMgmt, RewardMgmt 만들어야

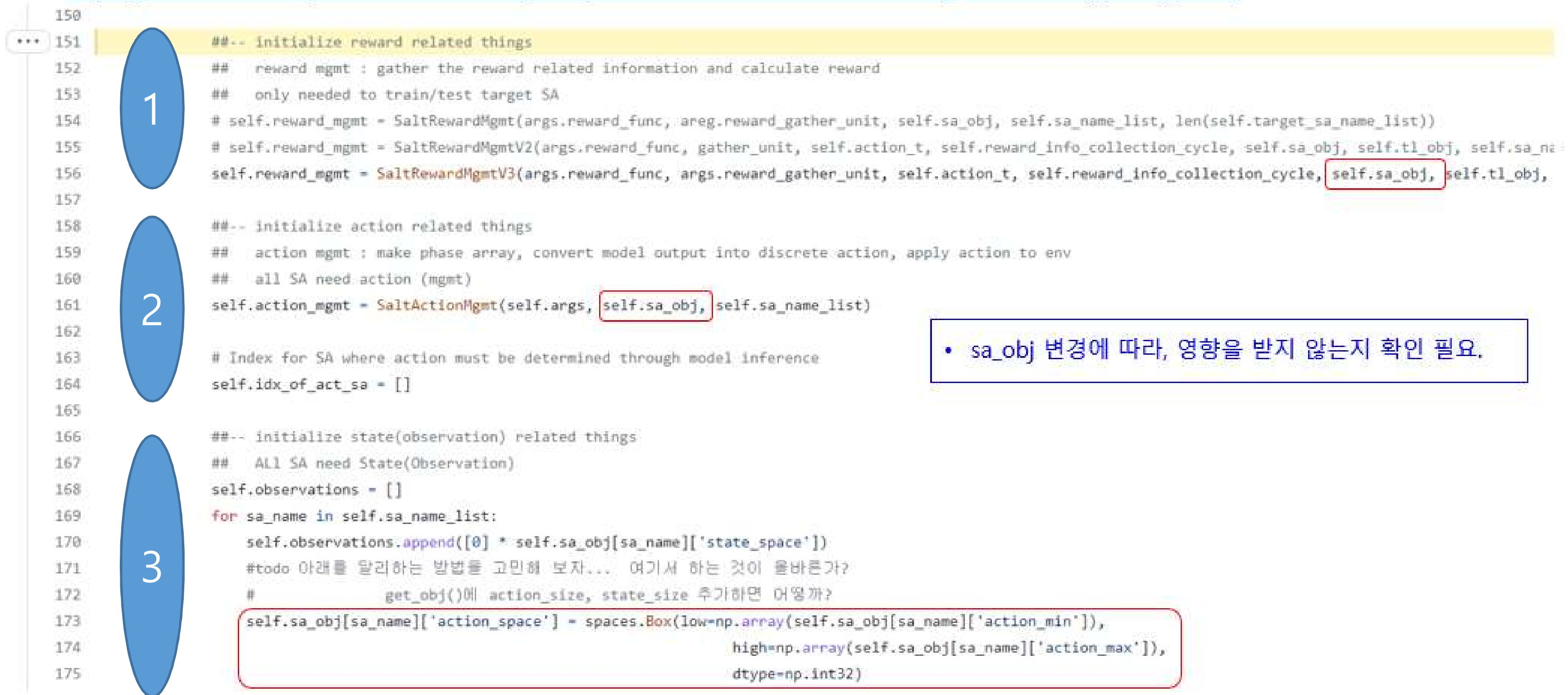
Env.sa_obj 객체 : : YJ.Lee's comment

SappoEnv.py #0

발췌 : Action Space 코드 수정 (v20230302, 이용진) 12p

sa_obj 객체 생성 후에 사용하는 부분

https://github.com/etri-city-traffic-brain/traffic-signal-optimization/blob/master/atsc-rl/multiagent_tf2/env/SappoEnv.py#L151



```python

##-- initialize reward related things
##   reward mgmt : gather the reward related information and calculate reward
##   only needed to train/test target SA
# self.reward_mgmt = SaltRewardMgmt(args.reward_func, areg.reward_gather_unit, self.sa_obj, self.sa_name_list, len(self.target_sa_name_list))
# self.reward_mgmt = SaltRewardMgmtV2(args.reward_func, gather_unit, self.action_t, self.reward_info_collection_cycle, self.sa_obj, self.tl_obj, self.sa_na
self.reward_mgmt = SaltRewardMgmtV3(args.reward_func, args.reward_gather_unit, self.action_t, self.reward_info_collection_cycle, self.sa_obj, self.tl_obj,

##-- initialize action related things
##   action mgmt : make phase array, convert model output into discrete action, apply action to env
##   all SA need action (mgmt)
self.action_mgmt = SaltActionMgmt(self.args, self.sa_obj, self.sa_name_list)

# Index for SA where action must be determined through model inference
self.idx_of_act_sa = []

##-- initialize state(observation) related things
##   ALL SA need State(Observation)
self.observations = []
for sa_name in self.sa_name_list:
    self.observations.append([0] * self.sa_obj[sa_name]['state_space'])
    #todo 아래를 달리하는 방법을 고민해 보자...  여기서 하는 것이 올바른가?
    #            get_obj()에 action_size, state_size 추가하면 어떨까?
    self.sa_obj[sa_name]['action_space'] = spaces.Box(low=np.array(self.sa_obj[sa_name]['action_min']),
                                                      high=np.array(self.sa_obj[sa_name]['action_max']),
                                                      dtype=np.int32)
```

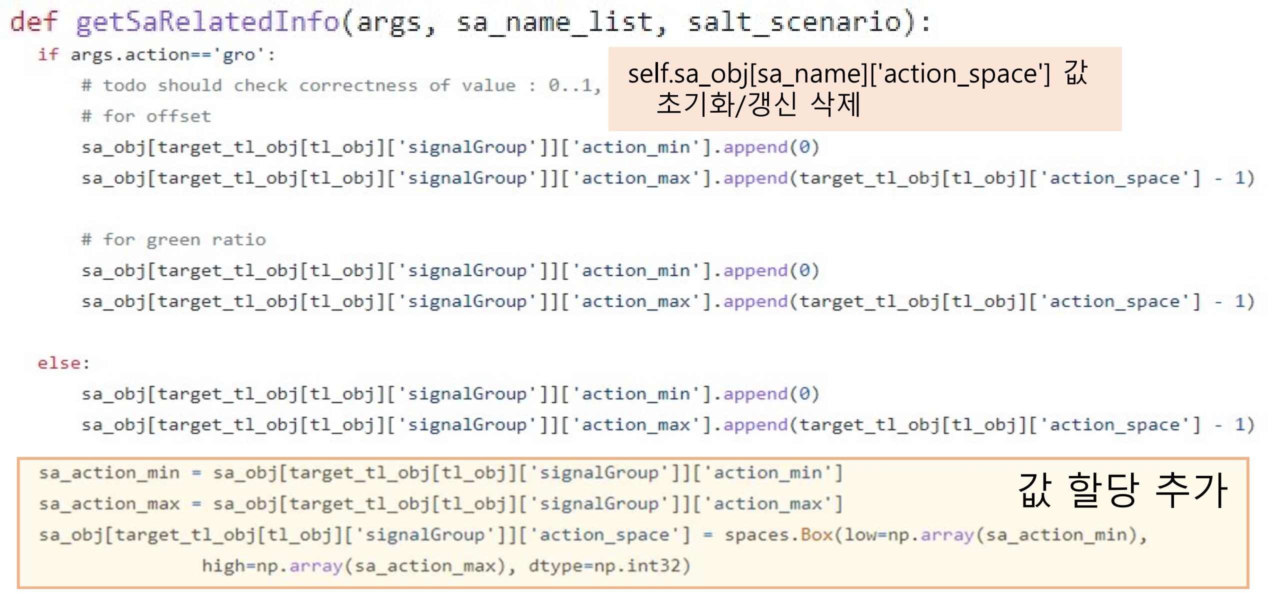
- sa_obj 변경에 따라, 영향을 받지 않는지 확인 필요.

안1 순서 변경 : 3 → 1 → 2
SA 별 action_space 설정 후
ActionMgmt, RewardMgmt 객체 생성

안2 sa_obj 생성시 관련 정보도 생성
getSaRelatedInfo()@SaltEnvUtil.py
(SaltEnvUtil.py, SappoEnv.py 수정)

Env.sa_obj 객체 : 수정

SaltEnvUtil.py



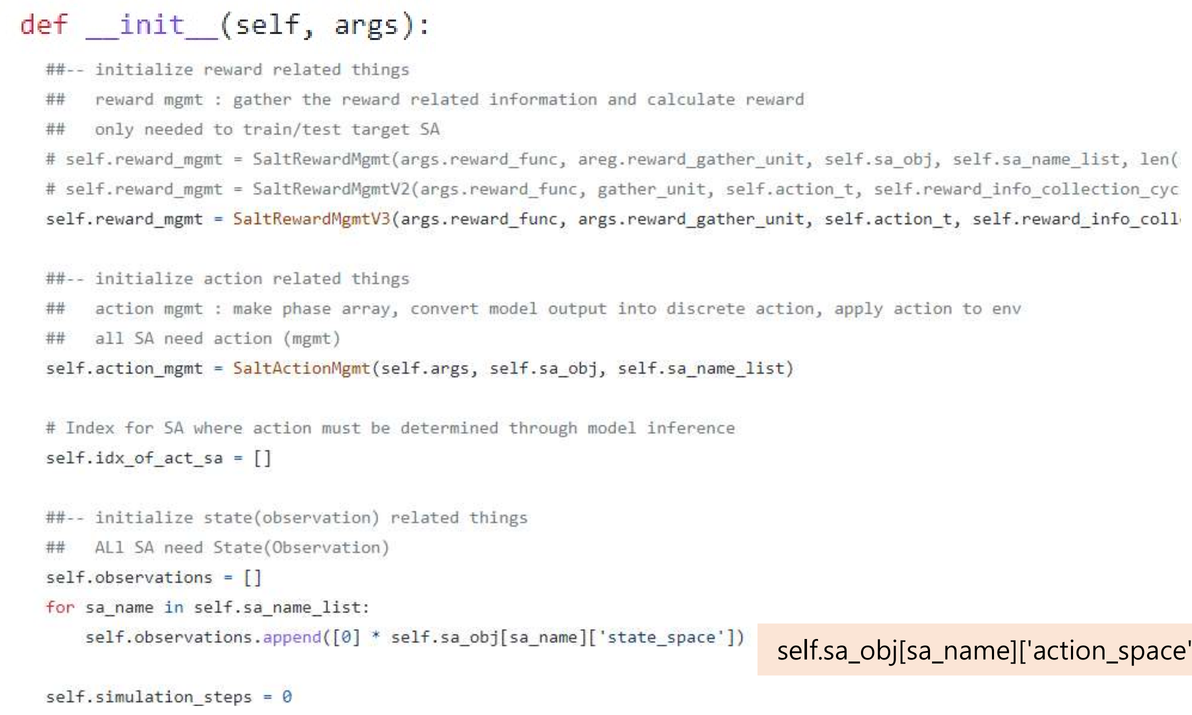
```python
def getSaRelatedInfo(args, sa_name_list, salt_scenario):
    if args.action=='gro':
        # todo should check correctness of value : 0..1,
        # for offset
        sa_obj[target_tl_obj[tl_obj]['signalGroup']]['action_min'].append(0)
        sa_obj[target_tl_obj[tl_obj]['signalGroup']]['action_max'].append(target_tl_obj[tl_obj]['action_space'] - 1)

        # for green ratio
        sa_obj[target_tl_obj[tl_obj]['signalGroup']]['action_min'].append(0)
        sa_obj[target_tl_obj[tl_obj]['signalGroup']]['action_max'].append(target_tl_obj[tl_obj]['action_space'] - 1)

    else:
        sa_obj[target_tl_obj[tl_obj]['signalGroup']]['action_min'].append(0)
        sa_obj[target_tl_obj[tl_obj]['signalGroup']]['action_max'].append(target_tl_obj[tl_obj]['action_space'] - 1)
```

self.sa_obj[sa_name]['action_space'] 값 초기화/갱신 삭제

```python
    sa_action_min = sa_obj[target_tl_obj[tl_obj]['signalGroup']]['action_min']
    sa_action_max = sa_obj[target_tl_obj[tl_obj]['signalGroup']]['action_max']
    sa_obj[target_tl_obj[tl_obj]['signalGroup']]['action_space'] = spaces.Box(low=np.array(sa_action_min),
                   high=np.array(sa_action_max), dtype=np.int32)
```

값 할당 추가

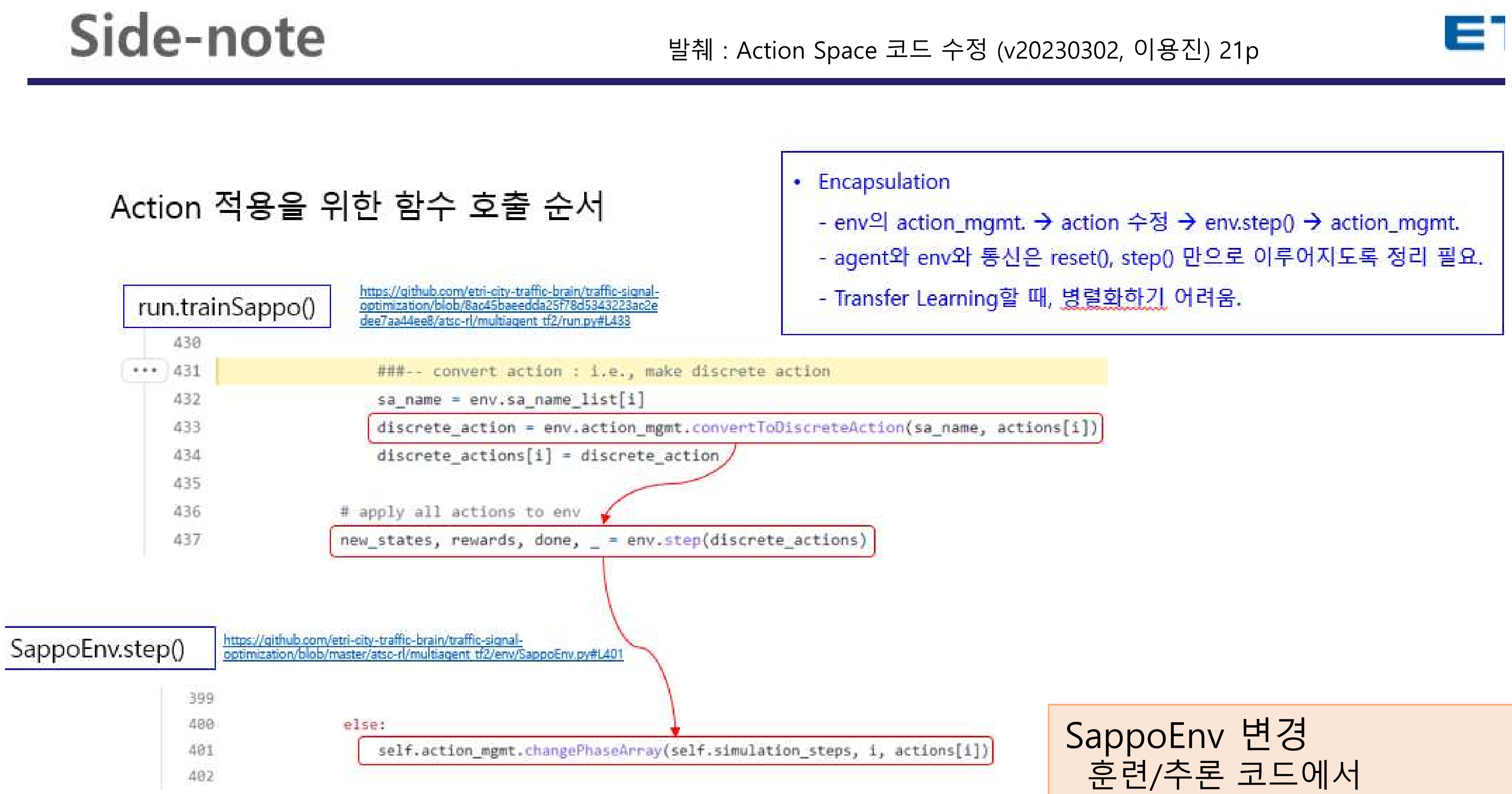
SaltEnv.py



```python
def __init__(self, args):
    ##-- initialize reward related things
    ##   reward mgmt : gather the reward related information and calculate reward
    ##   only needed to train/test target SA
    # self.reward_mgmt = SaltRewardMgmt(args.reward_func, areg.reward_gather_unit, self.sa_obj, self.sa_name_list, len(
    # self.reward_mgmt = SaltRewardMgmtV2(args.reward_func, gather_unit, self.action_t, self.reward_info_collection_cyc
    self.reward_mgmt = SaltRewardMgmtV3(args.reward_func, args.reward_gather_unit, self.action_t, self.reward_info_coll

    ##-- initialize action related things
    ##   action mgmt : make phase array, convert model output into discrete action, apply action to env
    ##   all SA need action (mgmt)
    self.action_mgmt = SaltActionMgmt(self.args, self.sa_obj, self.sa_name_list)

    # Index for SA where action must be determined through model inference
    self.idx_of_act_sa = []

    ##-- initialize state(observation) related things
    ##   All SA need State(Observation)
    self.observations = []
    for sa_name in self.sa_name_list:
        self.observations.append([0] * self.sa_obj[sa_name]['state_space'])

    self.simulation_steps = 0
```

self.sa_obj[sa_name]['action_space'] 값 할당 삭제

Env Encapsulation : YJ.Lee's comment



Side-note

발췌 : Action Space 코드 수정 (v20230302, 이용진) 21p

Action 적용을 위한 함수 호출 순서

run.trainSappo()

https://github.com/etri-city-traffic-brain/traffic-signal-optimization/blob/8ac45baeedda25f78d5343223ac2edee7aa44ee8/atsc-rl/multiagent_tf2/run.py#L433

```
430
431             ###-- convert action : i.e., make discrete action
432             sa_name = env.sa_name_list[i]
433             discrete_action = env.action_mgmt.convertToDiscreteAction(sa_name, actions[i])
434             discrete_actions[i] = discrete_action
435
436         # apply all actions to env
437         new_states, rewards, done, _ = env.step(discrete_actions)
```

SappoEnv.step()

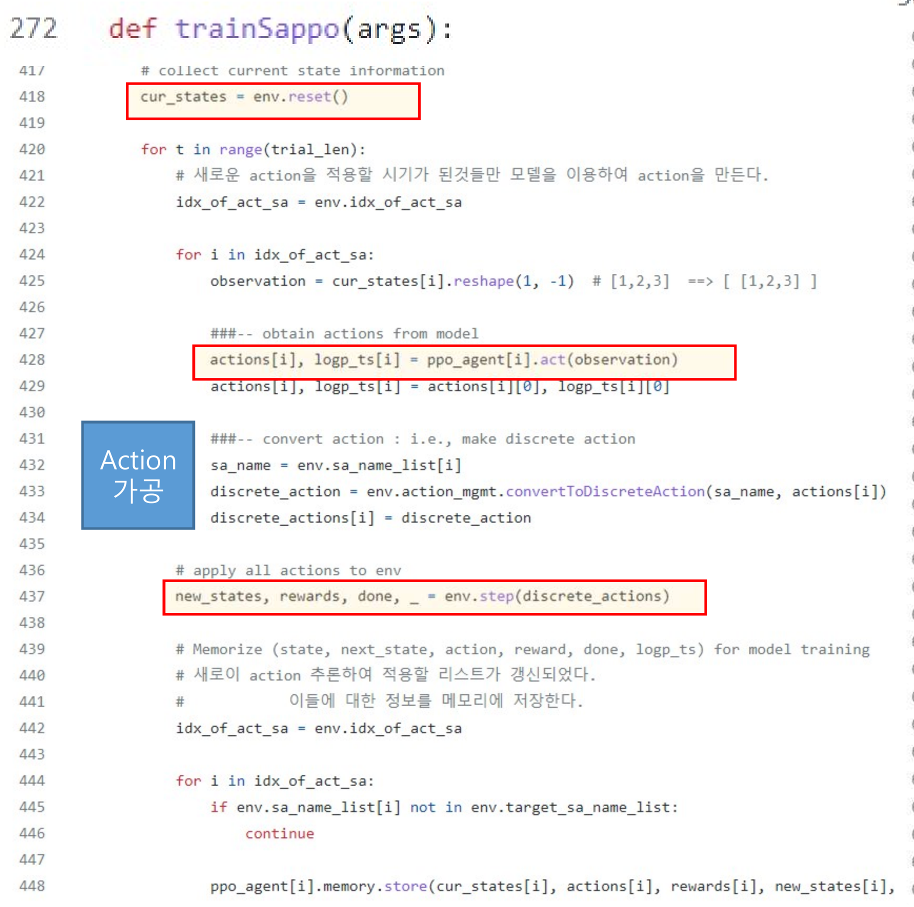
https://github.com/etri-city-traffic-brain/traffic-signal-optimization/blob/master/atsc-rl/multiagent_tf2/env/SappoEnv.py#L401

```
399
400         else:
401             self.action_mgmt.changePhaseArray(self.simulation_steps, i, actions[i])
402
```

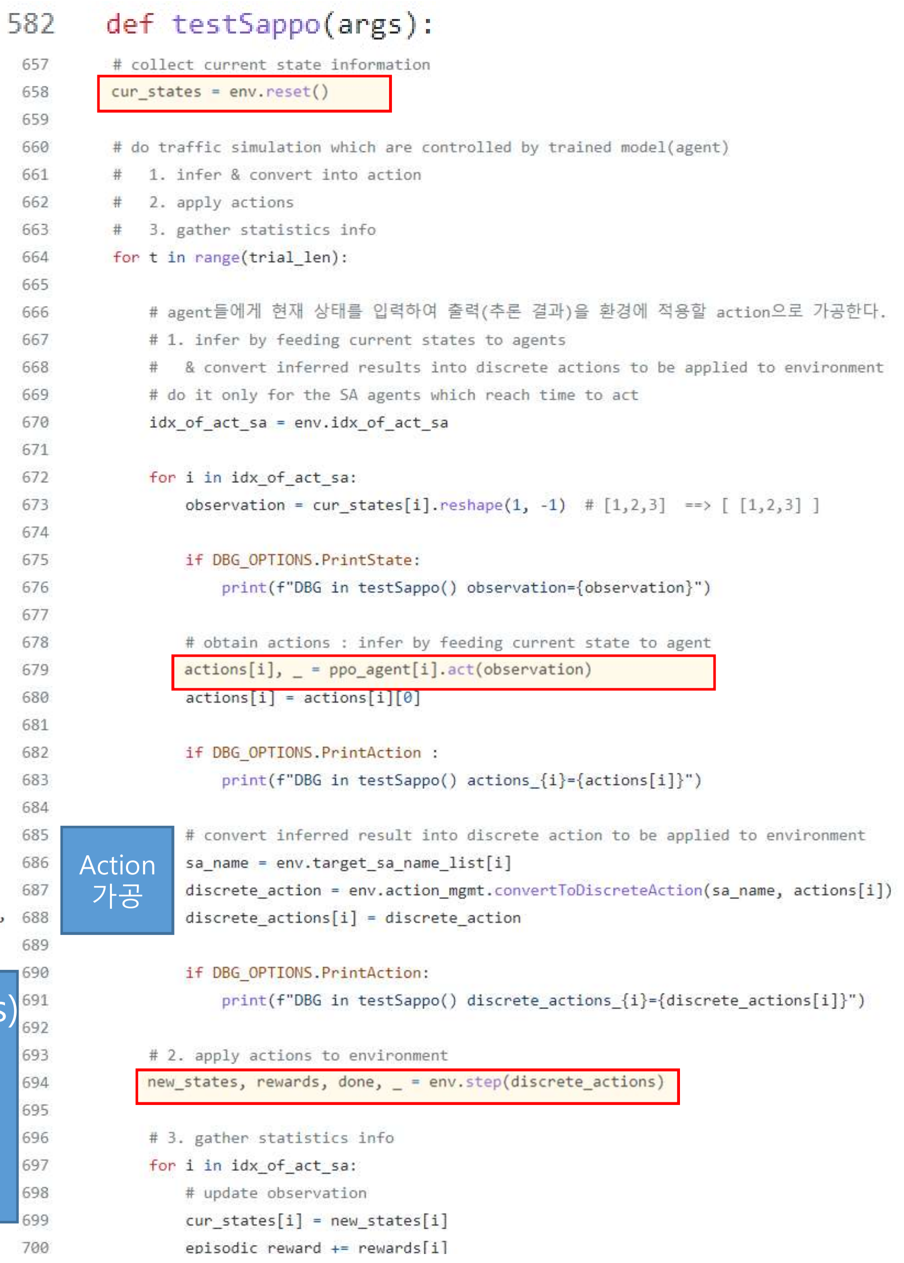
- Encapsulation
  - env의 action_mgmt. → action 수정 → env.step() → action_mgmt.
  - agent와 env와 통신은 reset(), step() 만으로 이루어지도록 정리 필요.
  - Transfer Learning할 때, 병렬화하기 어려움.

SappoEnv 변경
  훈련/추론 코드에서
  Env.reset(), Agent.getAction(),
  Env.step() 만 보일수 있도록 수정

Env Encapsulation : 현재



```python
def trainSappo(args):
    # collect current state information
    cur_states = env.reset()

    for t in range(trial_len):
        # 새로운 action을 적용할 시기가 된것들만 모델을 이용하여 action을 만든다.
        idx_of_act_sa = env.idx_of_act_sa

        for i in idx_of_act_sa:
            observation = cur_states[i].reshape(1, -1)  # [1,2,3]  ==> [ [1,2,3] ]

            ###-- obtain actions from model
            actions[i], logp_ts[i] = ppo_agent[i].act(observation)
            actions[i], logp_ts[i] = actions[i][0], logp_ts[i][0]

            ###-- convert action : i.e., make discrete action
            sa_name = env.sa_name_list[i]
            discrete_action = env.action_mgmt.convertToDiscreteAction(sa_name, actions[i])
            discrete_actions[i] = discrete_action

        # apply all actions to env
        new_states, rewards, done, _ = env.step(discrete_actions)

        # Memorize (state, next_state, action, reward, done, logp_ts) for model training
        # 새로이 action 추론하여 적용할 리스트가 갱신되었다.
        #             이들에 대한 정보를 메모리에 저장한다.
        idx_of_act_sa = env.idx_of_act_sa

        for i in idx_of_act_sa:
            if env.sa_name_list[i] not in env.target_sa_name_list:
                continue

            ppo_agent[i].memory.store(cur_states[i], actions[i], rewards[i], new_states[i],
```



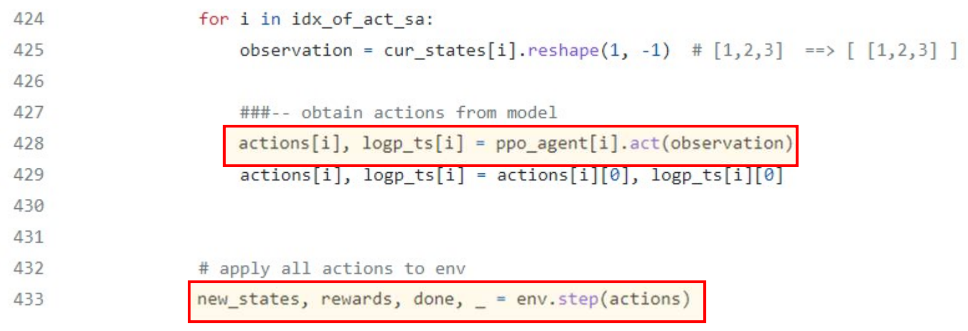
```python
def testSappo(args):
    # collect current state information
    cur_states = env.reset()

    # do traffic simulation which are controlled by trained model(agent)
    #   1. infer & convert into action
    #   2. apply actions
    #   3. gather statistics info
    for t in range(trial_len):

        # agent들에게 현재 상태를 입력하여 출력(추론 결과)을 환경에 적용할 action으로 가공한다.
        # 1. infer by feeding current states to agents
        #   & convert inferred results into discrete actions to be applied to environment
        # do it only for the SA agents which reach time to act
        idx_of_act_sa = env.idx_of_act_sa

        for i in idx_of_act_sa:
            observation = cur_states[i].reshape(1, -1)  # [1,2,3]  ==> [ [1,2,3] ]

            if DBG_OPTIONS.PrintState:
                print(f"DBG in testSappo() observation={observation}")

            # obtain actions : infer by feeding current state to agent
            actions[i], _ = ppo_agent[i].act(observation)
            actions[i] = actions[i][0]

            if DBG_OPTIONS.PrintAction :
                print(f"DBG in testSappo() actions_{i}={actions[i]}")

            # convert inferred result into discrete action to be applied to environment
            sa_name = env.target_sa_name_list[i]
            discrete_action = env.action_mgmt.convertToDiscreteAction(sa_name, actions[i])
            discrete_actions[i] = discrete_action

            if DBG_OPTIONS.PrintAction:
                print(f"DBG in testSappo() discrete_actions_{i}={discrete_actions[i]}")

        # 2. apply actions to environment
        new_states, rewards, done, _ = env.step(discrete_actions)

        # 3. gather statistics info
        for i in idx_of_act_sa:
            # update observation
            cur_states[i] = new_states[i]
            episodic_reward += rewards[i]
```

Env 객체에서 Action 가공하고 관리(discrete_actions)
- 멤버 변수로 유지
- reset()에서 초기화
- step()에서 가공 & 활용

➔ 수정 범위 최소화 (run.py, SappoEnv.py)

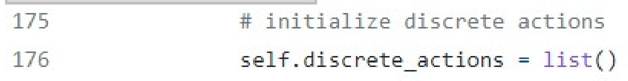
Env Encapsulation : 수정

trainSappo()/testSappo() at Run.py



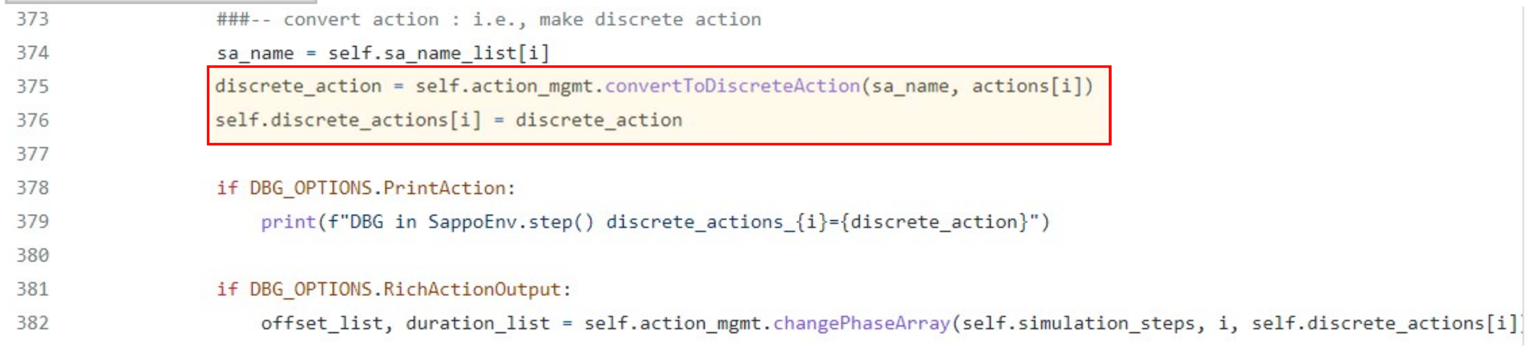
```
for i in idx_of_act_sa:
    observation = cur_states[i].reshape(1, -1)  # [1,2,3]  ==> [ [1,2,3] ]

    ###-- obtain actions from model
    actions[i], logp_ts[i] = ppo_agent[i].act(observation)
    actions[i], logp_ts[i] = actions[i][0], logp_ts[i][0]


# apply all actions to env
new_states, rewards, done, _ = env.step(actions)
```

SappoEnv::__init__()



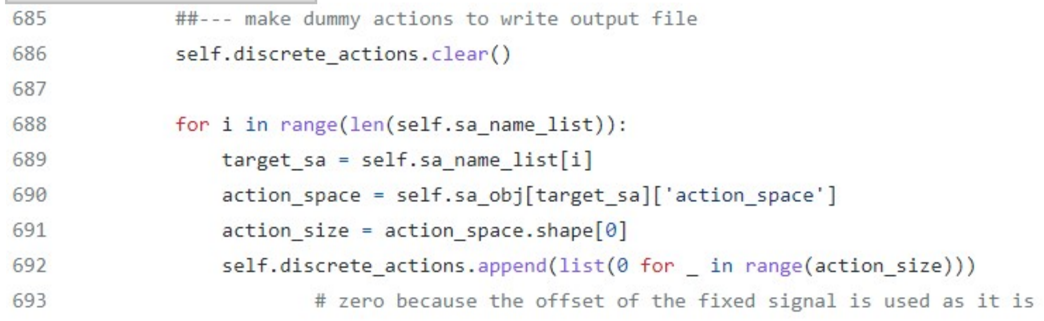
```
# initialize discrete actions
self.discrete_actions = list()
```

SappoEnv::step()



```
###-- convert action : i.e., make discrete action
sa_name = self.sa_name_list[i]
discrete_action = self.action_mgmt.convertToDiscreteAction(sa_name, actions[i])
self.discrete_actions[i] = discrete_action

if DBG_OPTIONS.PrintAction:
    print(f"DBG in SappoEnv.step() discrete_actions_{i}={discrete_action}")

if DBG_OPTIONS.RichActionOutput:
    offset_list, duration_list = self.action_mgmt.changePhaseArray(self.simulation_steps, i, self.discrete_actions[i]
```

SappoEnv::reset()



```
##--- make dummy actions to write output file
self.discrete_actions.clear()

for i in range(len(self.sa_name_list)):
    target_sa = self.sa_name_list[i]
    action_space = self.sa_obj[target_sa]['action_space']
    action_size = action_space.shape[0]
    self.discrete_actions.append(list(0 for _ in range(action_size)))
            # zero because the offset of the fixed signal is used as it is
```

Action의 신호 적용 : 모델 출력의 PhaseArray로 변환 개념

Phase array 생성

Duration = [10, 5]

InitialPhaseArray $\underbrace{0\ 0\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0\ 0}_{10}\ \ \underbrace{1\ 1\ 1\ 1\ 1}_{5}$

Offset 적용

Duration = [10, 5], offset = 4

InitialPhaseArray	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1
OffsetAppliedPA	1 1 1 1 0 0 0 0 0 0 0 0 0 0 1

녹색 시간 조정

Duration = [10, 5], offset = 4, action=[1, -1], add_time=1
➔ Duration = [10+1*1, 5+-1*1] = [11, 4]

InitialPhaseArray	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1
greenRatioAppliedPA	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
Offset 적용	1 1 1 1 0 0 0 0 0 0 0 0 0 0 0

Action의 신호 적용 : 모델 출력의 PhaseArray로 변환 주요 코드 설명(1/2)

```python
    def __getGreenRatioAppliedPhaseArray(self, curr_sim_step, an_sa_obj, actions):
        '''
        get green-ratio actions applied phase array list

        :param curr_sim_step: current sumulation step
        :param an_sa_obj: object which holds information about an SA
        :param actions: actions to apply
        :return:
        '''
        tlid_list = an_sa_obj["tlid_list"]
        # sa_cycle = an_sa_obj["cycle_list"][0]

        phase_sum_list = []
        phase_list = []
        phase_array_list = []

        if DBG_OPTIONS.RichActionOutput:
            duration_list=[]

        for tlid_idx in range(len(tlid_list)):
            tlid = tlid_list[tlid_idx]
            green_idx = an_sa_obj["green_idx_list"][tlid_idx][0]
            # min_dur = an_sa_obj["minDur_list"][tlid_idx]
            # maxDur = an_sa_obj['maxDur_list'][tlid_idx]
            currDur = an_sa_obj['duration_list'][tlid_idx]

            if DBG_OPTIONS.RichActionOutput:
                new_duration = currDur.copy()

            mpv = libsalt.trafficsignal.getCurrentTLSScheduleByNodeID(tlid).myPhaseVector
            mpv = list(mpv)
```

용도 없다?

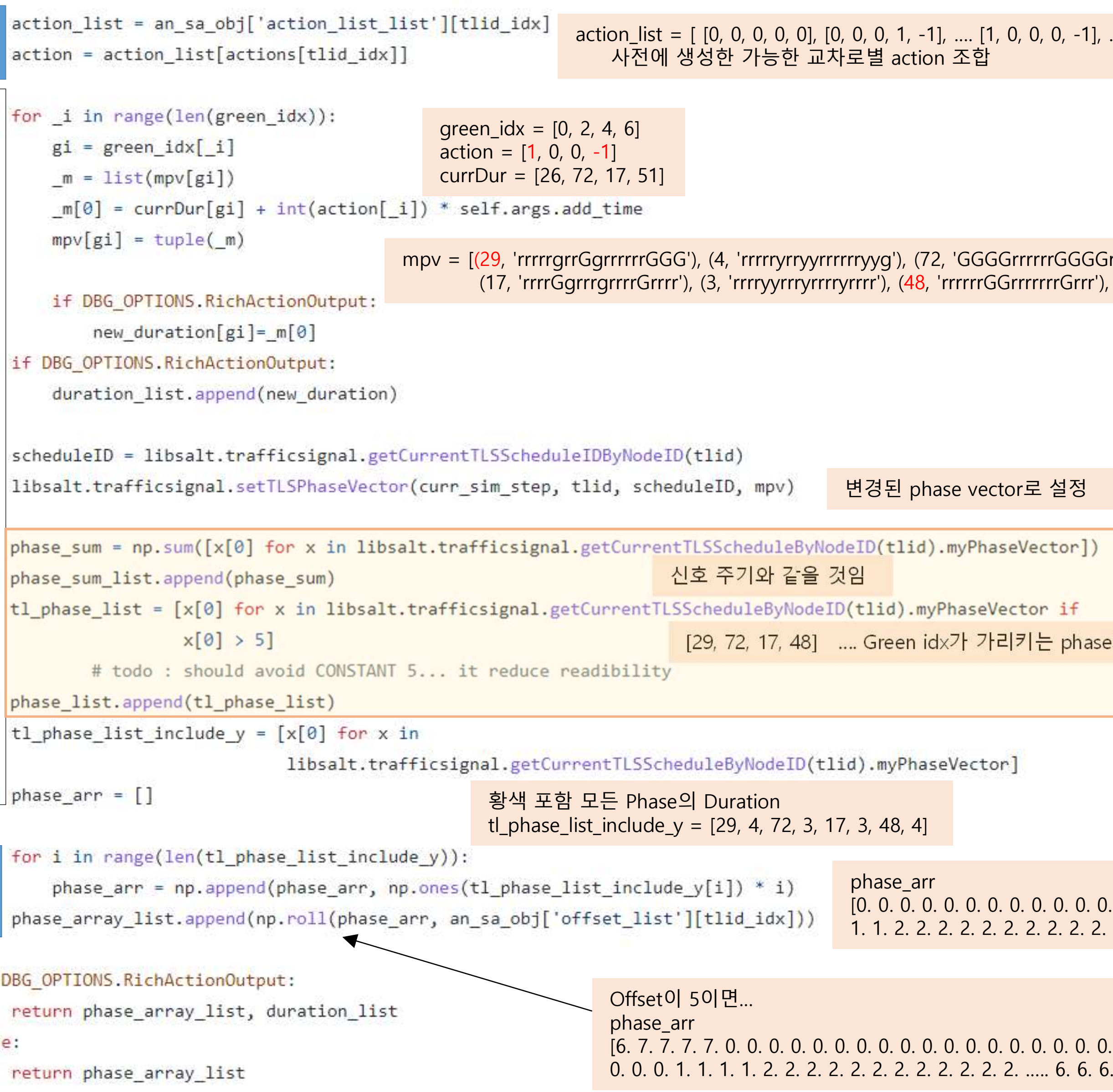
tlid = "cluster_563103641_563103889_563103894_563103895" # 원골 네거리

green_idx = [0, 2, 4, 6]
currDur = [26, 72, 17, 51]

[(26, 'rrrrrgrrGgrrrrrrGGG'), (4, 'rrrrryrryyrrrrrryyg'), (72, 'GGGGrrrrrrGGGGrrrrG'), (3, 'yyyyrrrrrryyyyrrrry'),
(17, 'rrrrGgrrrgrrrrGrrrr'), (3, 'rrrryyrrryrrrryrrrr'), (51, 'rrrrrrGGrrrrrrrGrrr'), (4, 'rrrrrryyrrrrrrryrrr')]

Action의 신호 적용 : 모델 출력의 PhaseArray로 변환 주요 코드 설명(2/2)

mpv = [(26, 'rrrrrgrrGgrrrrrrGGG'), (4, 'rrrrryrryyrrrrrryyg'), (72, 'GGGGrrrrrrGGGGrrrrG'), (3, 'yyyyrrrrrryyyyrrrry'), (17, 'rrrrGgrrrgrrrrGrrrr'), (3, 'rrrryyrrryrrrryrrrr'), (51, 'rrrrrrGGrrrrrrrGrrr'), (4, 'rrrrrryyrrrrrrryrrr')]



```
        action_list = an_sa_obj['action_list_list'][tlid_idx]
        action = action_list[actions[tlid_idx]]

        for _i in range(len(green_idx)):
            gi = green_idx[_i]
            _m = list(mpv[gi])
            _m[0] = currDur[gi] + int(action[_i]) * self.args.add_time
            mpv[gi] = tuple(_m)

            if DBG_OPTIONS.RichActionOutput:
                new_duration[gi]=_m[0]
        if DBG_OPTIONS.RichActionOutput:
            duration_list.append(new_duration)

        scheduleID = libsalt.trafficsignal.getCurrentTLSScheduleIDByNodeID(tlid)
        libsalt.trafficsignal.setTLSPhaseVector(curr_sim_step, tlid, scheduleID, mpv)

        phase_sum = np.sum([x[0] for x in libsalt.trafficsignal.getCurrentTLSScheduleByNodeID(tlid).myPhaseVector])
        phase_sum_list.append(phase_sum)
        tl_phase_list = [x[0] for x in libsalt.trafficsignal.getCurrentTLSScheduleByNodeID(tlid).myPhaseVector if
                                 x[0] > 5]
                # todo : should avoid CONSTANT 5... it reduce readibility
        phase_list.append(tl_phase_list)
        tl_phase_list_include_y = [x[0] for x in
                                   libsalt.trafficsignal.getCurrentTLSScheduleByNodeID(tlid).myPhaseVector]

        phase_arr = []

        for i in range(len(tl_phase_list_include_y)):
            phase_arr = np.append(phase_arr, np.ones(tl_phase_list_include_y[i]) * i)
        phase_array_list.append(np.roll(phase_arr, an_sa_obj['offset_list'][tlid_idx]))

    if DBG_OPTIONS.RichActionOutput:
        return phase_array_list, duration_list
    else:
        return phase_array_list
```

모델 출력을 적용할 action으로 가공

action_list = [[0, 0, 0, 0, 0], [0, 0, 0, 1, -1], ... [1, 0, 0, 0, -1], ...]
사전에 생성한 가능한 교차로별 action 조합

가공된 Action을

Phase duration에

적용

(sa_obj에 메달린 정보를 이용해도 될 것으로 생각됨.

단, 황색 정보를 포함한 모든 Phase Duration이 메달려 있어야 함)

green_idx = [0, 2, 4, 6]
action = [1, 0, 0, -1]
currDur = [26, 72, 17, 51]

mpv = [(29, 'rrrrrgrrGgrrrrrrGGG'), (4, 'rrrrryrryyrrrrrryyg'), (72, 'GGGGrrrrrrGGGGrrrrG'), (3, 'yyyyrrrrrryyyyrrrry'), (17, 'rrrrGgrrrgrrrrGrrrr'), (3, 'rrrryyrrryrrrryrrrr'), (48, 'rrrrrrGGrrrrrrrGrrr'), (4, 'rrrrrryyrrrrrrryrrr')]

변경된 phase vector로 설정

assert?

신호 주기와 같을 것임

[29, 72, 17, 48] .... Green idx가 가리키는 phase의 duration 과 같을 것임...

황색 포함 모든 Phase의 Duration
tl_phase_list_include_y = [29, 4, 72, 3, 17, 3, 48, 4]

Phase array 변환

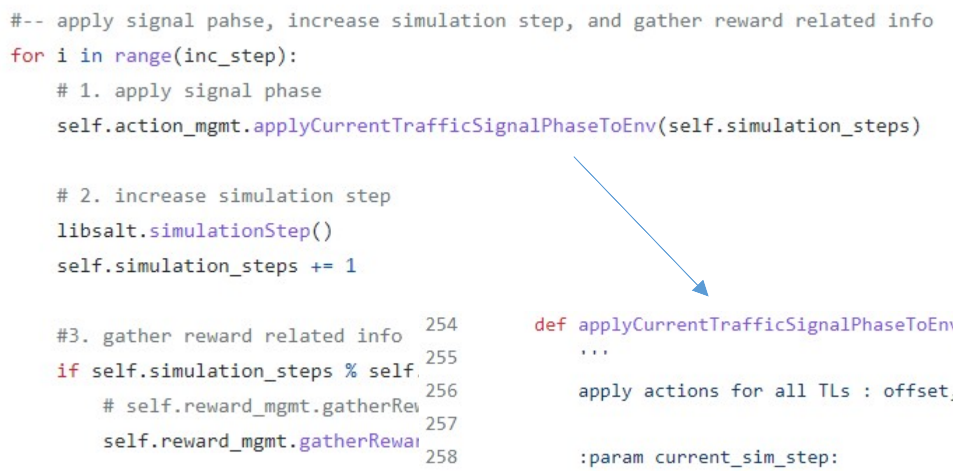
Offset 만큼 이동

phase_arr
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. ..... 6. 6. 6. 6. 6. 7. 7. 7. 7.]

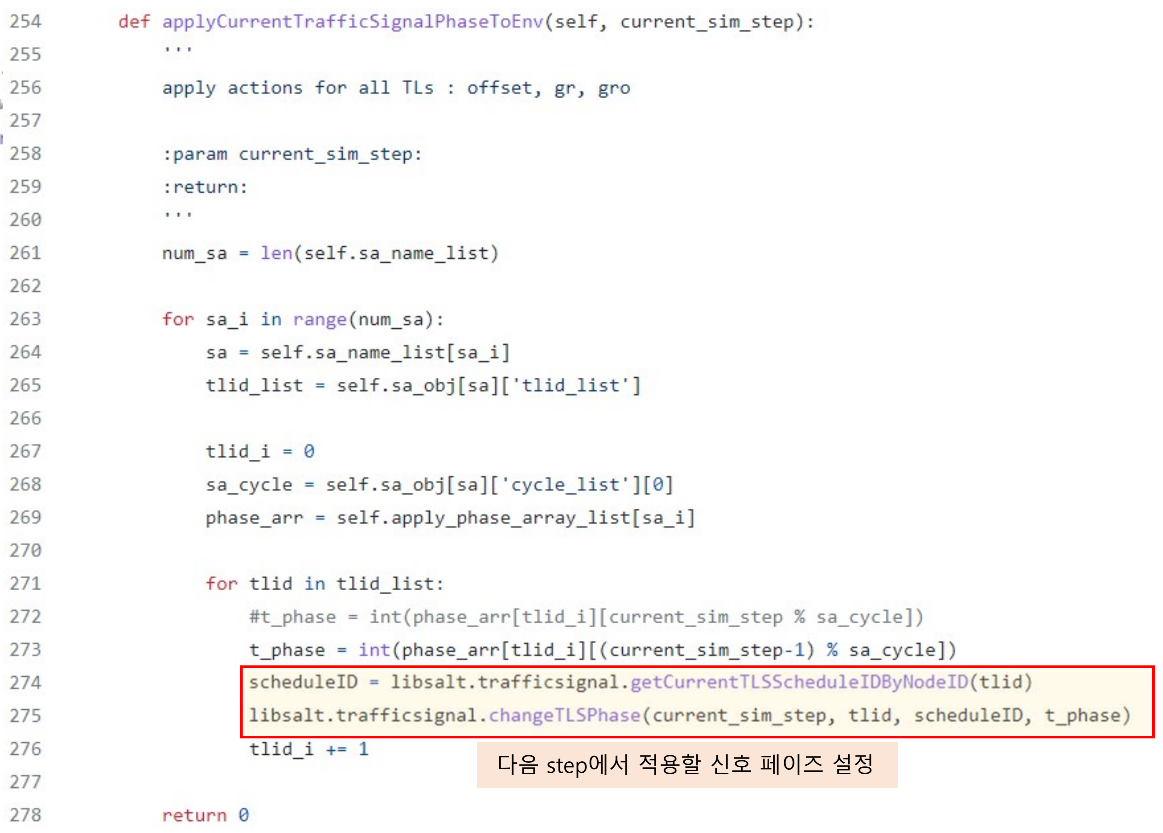
Offset이 5이면...
phase_arr
[6. 7. 7. 7. 7. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. ..... 6. 6. 6. 6. 6. 7. 7. 7. 7.]

Action의 신호 적용 : Phase Array 활용 변환된 신호 적용 코드 설명

SappoEnv:: Step()



```python
#-- apply signal pahse, increase simulation step, and gather reward related info
for i in range(inc_step):
    # 1. apply signal phase
    self.action_mgmt.applyCurrentTrafficSignalPhaseToEnv(self.simulation_steps)

    # 2. increase simulation step
    libsalt.simulationStep()
    self.simulation_steps += 1

    #3. gather reward related info
    if self.simulation_steps % self.
        # self.reward_mgmt.gatherRe
        self.reward_mgmt.gatherRewar
```



```python
def applyCurrentTrafficSignalPhaseToEnv(self, current_sim_step):
    '''
    apply actions for all TLs : offset, gr, gro

    :param current_sim_step:
    :return:
    '''
    num_sa = len(self.sa_name_list)

    for sa_i in range(num_sa):
        sa = self.sa_name_list[sa_i]
        tlid_list = self.sa_obj[sa]['tlid_list']

        tlid_i = 0
        sa_cycle = self.sa_obj[sa]['cycle_list'][0]
        phase_arr = self.apply_phase_array_list[sa_i]

        for tlid in tlid_list:
            #t_phase = int(phase_arr[tlid_i][current_sim_step % sa_cycle])
            t_phase = int(phase_arr[tlid_i][(current_sim_step-1) % sa_cycle])
            scheduleID = libsalt.trafficsignal.getCurrentTLSScheduleIDByNodeID(tlid)
            libsalt.trafficsignal.changeTLSPhase(current_sim_step, tlid, scheduleID, t_phase)
            tlid_i += 1

    return 0
```

다음 step에서 적용할 신호 페이즈 설정

Backup slides