2024 전기 졸업과제 중간보고서

강화학습 기반 V2G 전력거래 이익 최대화

지도교수 : 황원주

팀 명 : 노란전력

팀 원 : 201924488 배레온

202155517 김도균

목차

1. 연구 배경 및 목표

1.1) 연구 배경

온실가스 배출량의 지속적인 증가와 전례 없는 기상이변 들로 기후변화에 대한 위기의식이 고조되었다. 2018년에는 기후변화에 관한 정부 간 협의체 (Inter-governmental Panel on climate Change, IPCC)에서 2050년 까지 세계 탄소 순 배출량이 0이 되는 것을 목표로 삼고 있는 탄소 중립을 제안하였다. 2020년 1월 세계 경제 포럼에서는 세계가 직면한 가장 큰 위협으로 기후변화를 지목했으며 각국의 온실가스 감축 노력이 강화되어야 한다고 강조하였다. 게다가 2020년에는 선진국과 개도국이 함께 온실가스 배출 감축을 위해 맞은 파리협정 하에 신기후체제로 전환되었으며, 한국은 탄소 중립에 참가하였다. 한국은 초기에 2050 탄소중립 시나리오로 3개의 안을 공개하였다. 1안은 기존 체계와 구조를 최대한 활용하는 것이고 2안은 기술 발전으로 생활양식 변화를 3안은 화석연료 소비를 과감하게 줄이고 수소를 공급하는 방안이다. 이중 심의를 거쳐 확정된 2안과 3안에서는 재생에너지 비중이 매우 확대된다.

제주의 경우에는 2030 탄소 없는 섬으로 비전을 발표하며 탄소제로를 목표로 신재생 에너지 사용 비중을 전체 중 16% 이상이라는 높은 비율로 책정하여 에너지 정책을 시행하였고, 이에 따라 신재생 에너지 발전시설이 급증하게 되었다. 이에 따라 제주지역에서 수요에 비해 전력 공급이 많아지는 현상이 발생했으며, 풍력발전기 가동을 중단하는 등 출력제한 문제가 발생하였다.

호남 · 영남권 지역의 경우에는 땅값이 상대적으로 싸 태양광 발전설비가 급격하게 증가하였고, 전력이 과잉 공급되게 되었다. 하지만, 상대적으로 전력 수요가 높은 수도권으로 전력을 전송할 전력망 인프라가 부족한 상황으로 2023년 산업부는 공공기관이 보유한 설비를 우선으로 출력제한을 실시하였다. 이에 따라, 태양광 발전사업자들은 출력제한에 따른 금전적인 손실이 갈수록 늘어나고 있

다. 이러한 흐름은 신규 사업자들의 사업 의지를 꺾을 수 있으며, 신재생 에너지의 보급 확대에 문제가 될 수 있다.

1.2) 연구 목표

[그림 1]. 제주의 출력제한 해결 방안 (출처 : [1])

제주에서는 출력제한 해결을 위한 다양한 방안을 [그림 1]과 같이 내놓았다. 이중 가까운 미래에 적용하기 가장 적합한 것으로 보이는 대안은 미활용 전력 전기에너지를 저장하는 방안이다.

전력 과잉 공급은 한국만의 문제가 아니다. 미국의 재생에너지 생산량은 2020년 21%에서 2050년에 42%로 증가할 것으로 예상하였고, 필요 이상의 재생에너지가 생산되는 상황이 발생하였다. 이에 과다 생산된 전력을 저장하는 ESS 도입의 중요성이 대두되었다. 재생에너지의 생산 비중이 높은 캘리포니아는 2020년 기준 506MW의 ESS가 운영되고 있으며 추가로 1,027MW 규모의 도입을 준비하고 있다. 뉴욕은 2020년 기준 93MW 규모의 ESS를 도입하였으며, 1,076MW 규모의 ESS를 준비하고 있다.

중국은 2025년까지 ESS를 구축한 후, 본격적인 상용화에 접어들 것이라고 밝혔으며, 핵심 기술 및 장비의 자주화를 위해 중국 배터리 제조사인 CATL와 BYD에 유리한 조건을 내걸었다.

그 외에도 일본의 경우에는 대지진에 대비하여 비상 전원 확보 차원에서 ESS에 투자하고 있으며, 설치 보조금을 지원하고 세제

혜택을 주는 등의 정책을 실시하고 있다. 호주의 경우에는 16년의 대정전 이후 안정성 있는 국가 에너지 정책 추진을 위해 세계 최대 규모의 ESS를 구축하고 있으며, 프랑스는 태양광 발전 자가소비용 ESS 설치 자금을 지원하는 등의 정책을 실시하고 있다. 하지만, 이러한 ESS는 정작 한국에서는 많은 화재 사고와 재산 피해로 외면받고 있다.

이에 반하여 V2G는 전기차의 보급 확대에 따라 주목받는 시장으로 떠오르고 있다. 실제로 [그림 2]의 도표에 따르면, 국내 전기차 보급은 꾸준히 늘어나고 있으며, 전기차 충전 인프라 또한 매년 늘어나고 있다.

[그림 2]. 연도별 국내 전기차 · 충전기 보급 현황 (출처 : [2])

V2G는 전기차와 전력망을 양방향으로 연결하여, 전기차의 배터리를 전력망의 에너지원으로 활용하는 기술이며 이는 전기차가 단순한 이동 수단을 넘어, ESS의 역할을 할 수 있게 한다. V2G 시스템을 통해 전기차는 필요시 전력을 전력망에 공급하거나, 과잉 전력을 저장할 수 있는데 이러한 시스템은 사회 전반적 전력 비용 절감, 전력망의 안정성 유지, 에너지 관리의 최적화 등 지속 가능한 에너지 시스템을 운영하는 데 있어서 필수적이다.

하지만, 아직은 V2G 시스템을 통해 얻을 수 있는 경제적 이익이

충분히 크지 않으며 전기차 소유자가 V2G를 통해 얻는 금전적 혜택이 배터리 수명 단축으로 인한 비용을 상쇄할 만큼 충분하지 않은 경우가 많으므로 본팀은 V2G의 이익을 최대화할 수 있는 알고리즘을 개발하여 V2G가 사회 인프라에 적극적으로 도입될 수 있는 근거를 제공하고자 한다.

2. 요구조건 분석과 제약 사항

2.1) 문제 분석

2.1.1) 국내 ESS 시장의 위축

국내의 ESS는 연이은 화재 사고와 재산 피해로 많이 위축된 상황이다. ESS 운전 요금 지원의 종료로 시장 규모는 더욱 위축되었으며, 2021년까지의 손해액은 약 466억 원에 달한다. ESS 도입이 활발한 글로벌 시장과 달리 국내의 ESS는 외면받고 있다.

이에 반하여, 전기차 시장은 점차 확대되고 있으며, 충전기 또한 늘어나고 있다. 이에 전기차를 일종의 ESS로 사용하는 V2G를 도입하여 ESS와 연계하면, ESS의 수요 조절과 함께 ESS의 인식을 개선하고 편익도 늘리는 등의 다양한 이점을 기대한다.

2.1.2) 기존 스케줄 방식의 한계

전기차 시장의 경제 효율성을 위해 탄생한 양방향 충전 시스템은 현재 대부분 소규모 충전소의 EMS에서 고전적인 수학적 알고리즘 혹은 모델 예측 제어(MPC)를 통해 충전 스케줄을 생성한다. 이때 EMS는 전력비용을 최소화하며, ESS의 충전 및 방전 시점을 최적화해 전력망의 부하를 줄이는데 이러한 알고리즘은 충전소 규모가 작은 덜 복잡한 환경에서 최적화 문제를 해결하는 데 효과적이다. 하지만 도착하는 전기차, 출발하는 전기차 수, 전기차 배터리 용량, 충전 수요, 에너지 가격, 충전소의 ESS 개수, ESS 용량 등의 다양한 제약조건을 고려해야 하는 위 방식은 제약조건에 따라 EMS 프로그래밍은 어려움을 겪게 된다. 마찬가지로 충전소 규모가 커지거나 다른 요인으로 복잡한 환경의 충전소에서는 기존의 알고리즘을

채택하기 무척 어려워진다. 따라서 이러한 제약조건에 따른 복잡한 문제도 효율적으로 해결하며 다양한 조건 속에서 이익을 최대화할 방안을 모색하고자 한다.

2.2) 제약사항 및 대체안

개발한 알고리즘의 평가를 위하여 실제 전기차 충전소와 전기차를 활용하는 것은 현실적으로 어렵다. 이에, 시뮬레이션 환경을 활용하여 이익을 최대화할 방안을 찾고자 한다.

TABLE I
OVERVIEW OF EXISTING EV SIMULATORS FOCUSING ON SMART CHARGING STRATEGIES.

Simulator Name	V2G	Power Network Impact	EV Models	EV Behavior	Charging Stations	Available Baseline Algorithms	RL Ready	Programming Language	Comments
V2G-Sim [12]	Yes	Partial	Diverse	Real Charging Transaction Probability Distributions	Uniform	- Heuristics, Mathematical Programming	No	Not Open-source	- Customizable V2G simulations.
EVLibSim [13]	Yes	No	Diverse	Randomized	Uniform	- Heuristics, Mathematical Programming	No	Java	- Easily customizable simulations with a visual interface.
EV-EcoSim [14]	Yes	Complete	Uniform	Randomized	Uniform	- Mathematical Programming	No	Python	- Grid-Impact analysis
evsim [15]	No	Partial	Diverse	Real Charging Transaction Probability Distributions	Uniform	- Heuristics, Mathematical Programming	No	R	- Simulate and analyze the charging behavior of EV users.
OPEN [16]	Yes	Complete	Uniform	Randomized	Uniform	- Heuristics, Mathematical Programming	No	Python	- Modelling, control & simulations for smart local energy systems.
ACN-Sim [17]	No	Complete	Uniform	Real Charging Transactions	Uniform	- Heuristics, MPC, RL, Mathematical Programming	Yes	Python	- Designing a complete simulator framework.
SustainGym [18]	No	Complete	Uniform	Real Charging Transactions	Uniform	- RL	Yes	Python	- Providing a benchmark for sustainable RL applications.
Chargym [19]	Yes	Very Limited	Uniform	Randomized	Uniform	- Heuristics, RL	Yes	Python	- Comparing RL algorithms for smart charging.
EV2Gym (Ours)	Yes	Partial	Diverse	Real Charging Transaction Probability Distributions	Diverse	- Heuristics, MPC, RL, Mathematical Programming	Yes	Python	- Comprehensive simulator for any control algorithm.

[그림 3]. EV 시뮬레이터의 비교

[그림 3]은 현재 존재하는 다양한 시뮬레이션을 비교하는 자료다.

- V2G-Sim
  EV 모델과 동작과 같은 풍부한 기능을 갖췄지만, 오픈소스가 아니며 강화학습 개발을 위한 환경을 제공하지 않는다.
- EVLibSim
  다양한 EV 모델을 제공하고 있지만, Java로 작성되었으며 그리드에 미치는 영향을 시뮬레이션하지 않는다.
- EV-EcoSim
  자세한 배터리 사용률 및 성능 저하 모델이 포함되어 있으나,

현실적인 EV 사양 및 동작이 아닌 배터리에 중점을 두고 있다.

- EVsim
  EV 사용자의 행동을 평가하고 분석하는 데에 중점을 두었으며 사실적인 EV 행동 데이터로 구축되어 있으나, V2G의 영향을 조사할 수 있는 옵션이 포함되어 있지 않다
- OPEN
  EV를 포함하여 스마트 지역 에너지 시스템을 위한 통합 모델링, 제어 및 시뮬레이션을 위한 프레임워크로 배전 시스템 수준에서의 전력 흐름을 해결할 수 있다. 하지만, 통일된 EV 사양과 동작 모델만 있으며, 표준화된 Gym 환경이 없고 지나치게 단순화하여 강화학습 개발에 필요한 깊이가 부족하다.
- ACN-Sim
  표준화된 Gym 환경이 포함되어 있으며, EV 주차장 특성에 따라 개발되었다. 가장 확립된 EV 시뮬레이터 플랫폼 중 하나이며 다른 오픈소스 그리드 시뮬레이터를 사용하여 전력 네트워크 계산을 지원한다, 하지만, V2G 지원을 염두에 두지 않아 V2G 연구에는 부적합하다.
- SustainGym
  ACN-Sim에서 state와 action space를 정의하여 EV 최적 충전 문제를 강화학습 벤치마크로 표준화하였지만, 그 외의 실질적인 기능 추가가 없어 ACN-Sim과 같은 문제를 공유한다.
- Chargym
  비용 및 페널티 설계에 중점을 둔 강화학습 알고리즘을 개발하는 데 사용한다. 모든 EV와 충전기가 동일한 사양을 가지고 EV 동작이 실제 데이터를 기반으로 하지 않는 등 기본 모델이 매우 단순하여 강화학습 연구에 부적합하다.

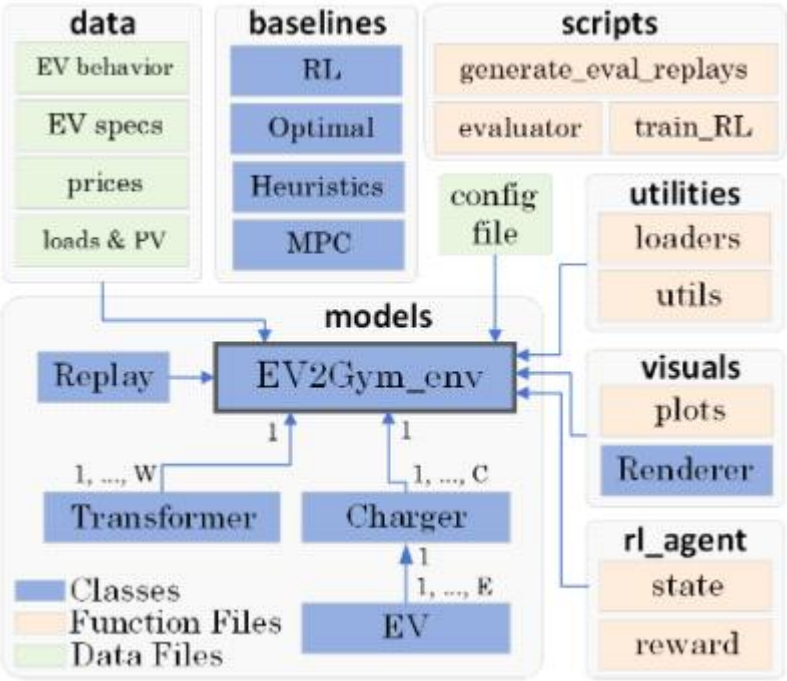
그 외에도 다양한 EV 충전 관리 시뮬레이션이 있지만, 분야가 다르거나 구식이라는 문제점이 있다. 기존 강화학습을 도입한 시뮬레이션보다 현실적이고 표준화된 시뮬레이션 환경이 필요하였고, 이

에 EV2Gym 시뮬레이션을 채택하였다.

3. 시뮬레이션 설계

3.1) EV2Gym [3]

EV2Gym은 V2G를 위한 오픈소스 시뮬레이터 환경이다. 강화학습 환경을 위한 파이썬 패키지 Gymnasium을 기반으로 구성되어 간소화된 강화학습 알고리즘 평가가 가능하다.



[그림 4]. EV2Gym 패키지의 폴더 및 파일 구조

[그림 4]는 EV2Gym 패키지의 폴더 및 파일 구조를 보여주며, 기본적인 클래스와 함수, 데이터 파일 등이 어떻게 상호 연결되어 있는지 보여준다. EV2Gym의 환경은 config file을 통해 정해진 변수로 이루어지며, Transformer, Charger, EV의 상호작용으로 동작한다. EV2Gym의 데이터는 실제 네덜란드의 오픈 데이터를 바탕으로 구성되어 있으며, EV의 행동과 스펙, 전기의 가격 등을 포함한다. 그 외에도 참고를 위한 baseline 알고리즘과 강화학습 state, reward 예시, 시각화를 위한 함수 등을 제공한다. 특히, EV2Gym은 오픈소스 겸 모듈식으로 필요에 따라 다양한 추가 기능을 추가 및 수정이 원활하게 가능하다는 장점이 있다.

Algorithm 1 Python code for running a simulation.

```
from EV2Gym.EV2Gym_env import EV2Gym
env← EV2Gym(config_file)                    ▷ Initialization
state, _ ← env.reset()
agent ← Algorithm(...)                      ▷ User-defined algorithm
for t = 1 to T do                           ▷ Simulation Phase
    actions ← agent.get_action(env, state)
    state, reward, done, _, stats ← env.step(actions)
end for
```

[그림 5]. EV2Gym의 실행 pseudocode

[그림 5]는 EV2Gym을 실행시키는 간단한 알고리즘을 보여준다. 시뮬레이션의 초기화 단계에서는 config file을 바탕으로 Charger, Transformer, EV의 개수 등 개별 시뮬레이션 환경의 다양한 매개변수를 설정한다. [표 1]은 config file에 포함된 세부적인 매개 변수를 보여준다.

[표 1] config file의 매개변수

구성 요소	매개 변수	기호
시뮬레이션	- 시간 간격 (분 단위)	Δt
	- 시뮬레이션 길이 (step 단위)	T
	- 시작 날짜와 시간	
	- 충전 토폴로지 및 속성	
	- EV 속성	
	- EV 시나리오 (주거용, 직장, 공용)	
	- EV 충전소의 집합	C
	- Power Transformer의 집합	W
EV	- AC 충전 전력의 최대, 최소 (kW)	$\underline{P}^{ch}_{AC}, \overline{P}^{ch}_{AC}$
	- DC 충전 전력의 최대, 최소 (kW)	$\underline{P}^{ch}_{DC}, \overline{P}^{ch}_{DC}$
	- 방전 전력의 최대, 최소 (kW)	$\underline{P}^{dis}, \overline{P}^{dis}$
	- 배터리 용량의 최대, 최솟값 (kWh)	$\underline{E}, \overline{E}$

	- 충전 및 방전 효율	η^{ch}, η^{dis}
	- 도착 시 배터리 용량 (kWh)	E^{arr}
	- 출발 시 원하는 배터리 용량 (kWh)	E^{*}
	- CV/CC 전환 SoC (%)	τ
	- 도착 시간 & 출발 시간 (t)	t^{arr}, t^{dep}
Charging Station	- 충전 스테이션 전류의 최대, 최소 (A)	$\underline{I}^{cs}, \bar{I}^{cs}$
	- EVSE 충전 전류의 최대, 최소 (A)	$\underline{I}^{ch}, \bar{I}^{ch}$
	- EVSE 방전 전류의 최대, 최소 (A)	$\underline{I}^{dis}, \bar{I}^{dis}$
	- 전압 (V) & 위상	V, ϕ
	- EVSE의 집합	J
	- 충전기의 종류 (AC 또는 DC)	
	- 충·방전 가격 (€ / kWh)	c^{ch}, c^{dis}
Transformer	- 전력의 최대, 최소 (kW)	$\underline{P}^{tr}, \bar{P}^{tr}$
	- Inflexible Loads (kW)	P^{L}
	- 태양광 발전 (kW)	P^{PV}
	- Demand Response Event (kW)	P^{DR}
	- 연결된 충전소 집합	C_w

3.2) 기타 사용 패키지

3.2.1) PyTorch

PyTorch는 신경망 구축에 사용되는 Python의 오픈소스 프레임워크다. 간단한 선형 회귀 알고리즘부터 복잡한 신경망까지 지원하며, 유연성과 사용 편의성 등의 이점으로 많은 연구에서 사용하는 머신러닝 프레임워크다. 이 과제에서는 강화학습 알고리즘의 구현에 사용하고자 한다.

3.2.2) Gymnasium

Gymnasium은 강화학습을 위한 환경을 제공하는 Python 오픈소스 라이브러리로 기존의 Open AI에서 제공하던 Gym의 포크 버전

이다. Gymnasium은 강화학습 알고리즘의 개발, 테스트, 비교를 위해 설계되어 다양한 알고리즘을 평가할 수 있는 환경을 제공해 준다. 이 과제에서 사용하는 EV2Gym 시뮬레이션 또한 Gymnasium 기반의 오픈소스 라이브러리이다.

3.2.3) Matplotlib

Matplotlib는 데이터의 시각화를 도와주는 Python의 라이브러리이다. numpy와 pandas의 시각화를 지원하며, 다양한 방식으로 많은 그래프를 그릴 수 있도록 도와준다. 이 과제에서는 알고리즘의 평가 결과를 시각화할 때 사용하고자 한다.

3.2.4) Stable_baselines3

Stable_baselines3는 OpenAI Baselines를 기반으로 강화학습 알고리즘을 구현한 라이브러리이다. 단순한 사용법으로 초보자도 구현 없이 강화학습 알고리즘을 평가할 수 있도록 도와주며, 세부적인 설정이 가능하여 상급자에게는 다양한 접근법을 비교할 수 있도록 도와준다. 이 과제에서 사용하는 강화학습 알고리즘은 모두 코드로 구현하는 것을 목표로 하나, TRPO, SAC와 같은 수학적 이론이 복잡한 경우 참고용으로 사용하고자 한다.

3.2.5) Gurobi

최적화 문제를 풀기 위한 솔버 중 하나로, 혼합 정수 선형 및 2차 최적화 문제를 해결할 수 있는 Python 라이브러리이다. 이 과제에서는 MPC 알고리즘의 구현에 사용하고자 한다.

4. 연구 내용

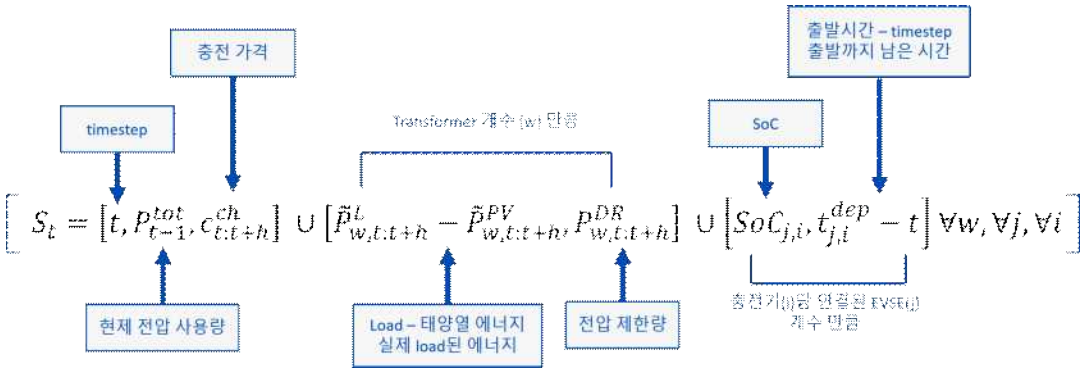
4.1) state, action, reward 설계

4.1.1) V2G Profit Maximization with Loads

문제는 EV2Gym에서 예시로 사용된 V2G Profit Maximization with Loads를 기반으로 정의한다. 충전기에 연결된 EV의 출발시간과 목표로 하는 SoC가 제공되며, 충전 에너지의 가격과

Transformer의 부하와 PV의 발전량 또한 제공되는 것을 가정한다. 목표는 EV 사용자의 요구와 V2G의 이익을 최대한으로 가져가면서, Transformer의 과부하를 방지하는 것이다.

4.1.2) state



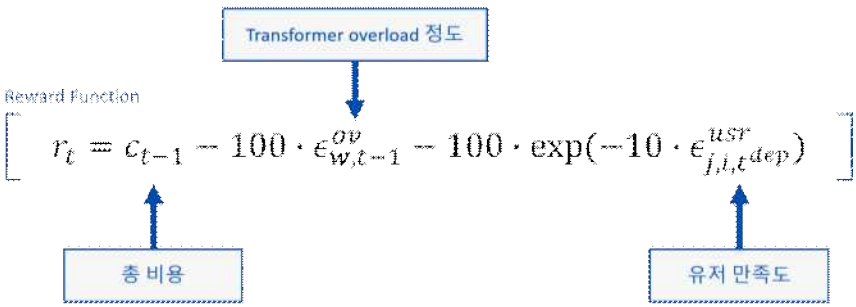
[그림 6]. State Function의 구조

State는 시뮬레이션의 상태를 나타내는 값으로 필요한 정보를 담은 vector이다. vector size는 $1+h+2hW+2N$ 로 h는 horizon, W는 Transformer의 개수, N은 EVSE의 개수를 의미한다.

4.1.3) action

action은 각 EVSE의 충전량을 바탕으로 계산되며, 강화학습 중 제한된 동작을 보장하기 위해 [-1, 1]의 범위로 제한하여 계산한다. 각각, 1은 최대로 충전, -1은 최대로 방전을 의미하며, 총 EVSE의 개수만큼의 vertor space를 가진다.

4.1.4) reward



[그림 7]. Reward Function의 구조

reward는 해당 action으로 얻을 수 있는 비용, 유저 만족도, Transformer의 overload 정도로 계산한다. 유저 만족도는 유저가 목표로 하는 SoC상태와 현제 SoC 상태의 비율로 계산하며, Transformer overload 정도는 Load된 Power량이 limit Power를 넘은 값 만큼을 사용한다.

4.2) 알고리즘 선정

4.2.1) Heuristics

- AFAP (As Fast As Possible)
  연결 직후 EV를 최대 전력으로 충전한다.
- ALAP (As Late As Possible)
  SoC에 도달하기 위해 가능한 한 늦게 EV 충전을 시작한다.
- RR (Round Robin)
  전력 설정값까지만 EV를 차례로 충전하여 각 EV가 주기적으로 공정하게 에너지 분배를 받는다.

Heuristics는 방전을 고려하지 못하기에 방전 유무에 따른 비교 용으로 사용하고자 한다.

4.2.2) MPC (Model Predictive Control) [4]

MPC는 Optimal Control의 한 방법이다. MPC의 핵심 원칙은 시스템의 수학적 모델에 있다. 즉, 제대로 된 모델링과 예측할 수 있는 역학 안에서 성능을 최대화하는 제어 작업을 생성하는 것이다. 이는 강화학습과 유사한 방식으로 기존의 Planning 문제에서 많이 사용되었기에 강화학습의 비교 대상으로 사용하고자 한다.

4.2.3) A2C (Advantage Actor Critic) [5]

구글 딥마인드에서 제안된 A3C (Asynchronous Advantage Actor Critic)의 Synchronous 변형 알고리즘이다. Actor-Critic 방식을 기반으로 하며, Advantage라는 개념을 도입하여, 예상 보상과 현재 가치 함수 간의 차이로 액터 정책을 업데이트하는 것이 특징이다.

여기서 Actor는 환경의 state를 기반으로 action을 정하는 역할을 한다. Actor는 Critic으로부터 피드백을 받아 최적의 Policy를 찾는 것을 목표로 한다. Critic은 Actor가 정한 action의 가치를 평가한다. 미래 보상의 총합과 가치함수를 계산하여 Actor의 성능을 평가한다.

Advantage는 Critic이 예측한 가치와 실제 보상 간의 차이로 계산되며, Actor가 정한 Action이 평균보다 얼마나 더 좋거나 나쁜지 판단하는 기준이 된다. 이를 통해 기존의 Actor-Critic보다 더 효율적인 학습을 가능하게 하는 것이 A2C 알고리즘의 특징이다.

4.2.4) DDPG (Deep Deterministic Policy Gradient) [6]

DDPG 또한 구글 딥마인드에서 제안된 알고리즘으로, 기존의 DPG (Deterministic Policy Gradient) 알고리즘에서 DQN (Deep Q-Network) 알고리즘을 접목했다. DDPG는 DQN에서 나온 좋은 결과 등을 연속 Action 영역에서도 사용할 수 있도록 확장하고자 하였다.

기존, DQN 이산적이며 차원이 낮은 action space에서만 한정적으로 사용할 수 있었기에 action의 종류가 늘어날수록 action space의 크기가 기하급수로 늘어나는 차원의 저주 문제가 있었다. 이에 DDPG에서는 이 문제를 DPG를 접목하여 해결하였다. 즉, DQN의 신경망을 사용한 학습과 DPG의 연속 환경에서 가능한 학습을 결합한 알고리즘이다.

DDPG의 특징으로는 Replay Buffer와 Target Network, Action Noise로 나눌 수 있다.

Replay Buffer는 DQN에서 가져온 것으로 모든 state를 step에 저장하고 실제 학습은 replay buffer에서 무작위로 뽑아 쓰는 방식이다. 기존의 어느 정도 순서가 정해져 있는 데이터의 무작위 성과 균일성이 떨어진다는 단점을 해결하였다.

DDPG의 Target Network는 Actor-Critic을 기반으로 한다. 즉, Actor-Critic이 총 2개로 4개의 신경망을 사용한다. Target

Network를 서서히 업데이트하며, Critic은 Taget과 자신의 예측치를 줄이는 방식으로, Actor는 Critic의 감정값을 극대화하는 방식으로 학습하게 된다.

DDPG는 Exploration을 위해 Action noise를 추가해 준다. 이는 action에 무작위 값을 추가해 주는 방식으로, ε-Greedy 방식을 사용한다.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

[그림 8]. DDPG pseudocode

4.2.5) TRPO [7]

TRPO는 정책 경사도 방법을 효율적으로 사용하기 위해 제안되었다. 정책 경사도에서 step-size는 중요한 역할을 하는데, 정책 경사도가 너무 커지면, 과장된 향상 방향을 따라 발산할 위험이 있고, 너무 작아지면, 학습 속도가 느려져 학습 효율이 떨어지는 문제가 있다.

TRPO는 정책 향상이 보장되는 이론적인 알고리즘을 바탕으로 근사하여 최적 정책으로 수렴시킨다. 이때, 본래의 목적에 따라 step-size를 KL divergence로 제한한다. 하지만, 모든 state에 대

한 KL divergence 계산을 하는 것은 현실적으로 어려움이 있기에 몬테카를로 기법을 이용하여 계산하게 된다. 즉, 2차 미분을 통해 Optimize가 이루어진다.

4.2.6) PPO [8]

PPO는 TRPO와 같은 목적을 가지고 있다. 주어진 데이터를 가지고 현재 policy를 최대한 큰 step만큼 향상하면서도 발산할 정도로 크게는 업데이트하지 않는 것이 목적이다. PPO와 TRPO의 차이점은 Penalty Term 즉, 너무 크게 변경되지 않도록 제한하는 부분에서 차이를 보인다. TRPO처럼 KL divergence를 사용하는 대신에 Clipping 방식을 사용하였다. 이를 통해 TRPO와 같은 방식의 계산을 2차 미분이 아닌 1차 미분으로 계산할 수 있도록 하면서, 비교적 간단한 구현으로 실용적으로 사용할 수 있게 되었다.

Algorithm 1 PPO, Actor-Critic Style

for iteration=$1, 2, \ldots$ **do**
  for actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

[그림 9]. PPO pseudocode

4.2.7) SAC [9]

SAC는 실제 환경에서 Model-free 알고리즘 적용이 어렵다는 문제에서 시작되었다. On-policy의 경우에는 갱신마다 샘플링 과정이 필요하기에 샘플 효율이 저하된다는 문제가 있고, Off-Policy의 경우에는 연속적인 state와 action에서 샘플의 복잡도가 높아진다는 문제가 있다. DDPG의 경우 이를 Off-policy에 정책 경사도 기반으로 해결했으나, 하이퍼파라미터 변화에 대한 민감성이 높고 수렴성이 약하다는 문제가 있었다. 이에 SAC는 Off-policy와 정책 경사도 기반을 따르면서, 하이퍼파라미터의 민감성이 낮고 수렴성이 높은 알고리즘을 제안하였다.

SAC는 기존 강화학습의 목표인 기대 보상의 최대화에 엔트로피 최대화를 사용하였다. 이를 통해 확률적인 최적 정책을 구성할 수 있도록 하였다. 또한, 엔트로피 최대화가 적용된 정책 반복알고리즘인 SPI (Soft Policy Iteration)를 기반으로 동작하는데, SPI의 요구 연산량이 많고 Table 방식에서만 적용 가능하다는 단점을 근사화를 통해서 해결하였다.

Algorithm 1 Soft Actor-Critic

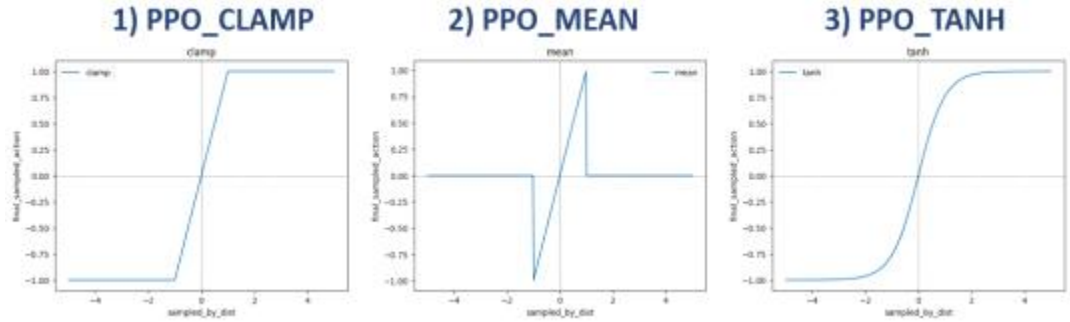
Initialize parameter vectors $\psi, \bar{\psi}, \theta, \phi$.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
  end for
end for

[그림 10]. SAC pseudocode

4.3) Action Sampling

4.3.1) Action Sampling

PPO 알고리즘의 Actor가 예측을 할때, 강화학습의 탐험을 보장하기 위해서, 정확한 값을 사용하는 것이 아닌, Gaussian 분포하에 sampling된 값을 사용한다. 이때, Actor가 예측한 값은 원하는 action의 분포인 [-1, 1]을 만족하지만, Gaussian 분포를 거치면, 보장할 수 없게 된다. 이로 인하여, 실제로 11% 정도의 범위를 벗어난 action이 나오게 되었고, 이에 다른 Action Sampling 방법을 사용해 보았다.



[그림 11]. Action Sampling 방법별 그래프

4.3.2) Clamping 사용

해당 방법은 범위를 벗어난 값을 범위안의 값으로 제한시키는 clamping 함수를 사용한 것이다. 가장 구현이 단순하면서, 기존의 값을 보존할 수 있는 방법이다.

4.3.3) Mean 값 사용

해당 방법은 Gaussian 분포상 가장 sampling 될 확률이 높은 mean 값을 사용하는 방식이다. 확률 분포를 가장 잘 보존할 수 있는 방법이다.

4.3.4) Tanh 함수 사용

해당 방법은 Tanh 함수의 결과가 [-1, 1] 사이라는 점을 사용하여, sampling 된 값이 Tanh 함수를 한번 더 거치도록 한 방식이다. 급격한 변화를 주는 Clamping, Mean과는 다르게 부드럽게 치환할 수 있는 방법이다.

4.4) 시뮬레이션 결과

[표 2]. V2G Profit Maximization 충전기 10개 (1000 에피소드)

Env	Algorithm	Profits/costs ₵	ϵ^{USF} (%)	Energy Ch.(kWh)	Energy Disch.(kWh)	Tr. Ov.(kWh)	Execution Time(s)	Reward ($\times 10^{-3}$)
Env1	PPO	7.60	65.86	74.56	91.32	42.69	0.05	-4.30
	PPO_CLAMP	17.66	53.53	48.94	110.11	31.10	0.05	-3.15
	PPO_MEAN	9.89	62.61	63.82	92.35	29.63	0.04	-3.00
	PPO_TANH	22.83	49.16	56.42	134.75	9.77	0.05	-1.01
	SAC	16.08	56.36	73.23	123.65	31.26	0.02	-3.16
	TRPO	24.75	48.25	65.34	145.06	8.08	0.02	-0.83
Env2	PPO	8.84	64.12	69.84	92.67	40.15	0.05	-4.04
	PPO_CLAMP	17.76	53.41	49.59	111.23	34.29	0.05	-3.47
	PPO_MEAN	10.03	62.39	63.22	92.72	31.42	0.05	-3.18
	PPO_TANH	24.70	46.40	48.69	136.63	19.44	0.05	-1.98
	SAC	15.50	56.98	75.10	123.11	29.03	0.02	-2.94
	TRPO	25.63	46.66	59.28	144.35	8.13	0.02	-0.84

[표 3]. V2G Profit Maximization 충전기 20개 (1000 에피소드)

Env	Algorithm	Profits/costs ₵	ϵ^{USF} (%)	Energy Ch.(kWh)	Energy Disch.(kWh)	Tr. Ov.(kWh)	Execution Time(s)	Reward ($\times 10^{-3}$)
Env1	PPO	51.49	43.93	67.69	258.57	231.99	0.06	-23.28
	PPO_CLAMP	37.17	52.41	95.98	225.90	43.19	0.06	-4.39
	PPO_MEAN	51.78	43.86	66.28	260.57	83.50	0.06	-8.43
	PPO_TANH	33.17	54.16	95.81	210.41	62.66	0.06	-6.34
	SAC	6.14	72.08	222.08	203.01	135.64	0.03	-13.62
	TRPO	9.92	67.84	151.00	162.67	73.74	0.02	-7.43
Env2	PPO	51.50	43.93	67.60	258.50	160.68	0.06	-16.15
	PPO_CLAMP	36.99	52.45	93.70	223.39	31.85	0.06	-3.26
	PPO_MEAN	51.71	43.67	65.80	261.44	39.73	0.06	-4.06
	PPO_TANH	32.96	54.24	96.10	210.01	48.31	0.06	-4.90
	SAC	6.59	73.24	252.86	224.68	177.95	0.03	-17.84
	TRPO	9.03	68.39	165.05	172.67	73.18	0.02	-7.37

가장 좋았던 리워드는 파란색, 그 다음으로 좋았던 리워드는 초록색으로 표시하였다. 대부분의 경우에서는 TRPO가 가장 좋은 성능을 보였으나, 충전기의 개수가 늘어나면 늘어날수록 즉, 문제가 복잡해질수록 TRPO 보단 Action Sampling 방식을 변형한 PPO들의 성능이 좋았다. Action Sampling 방식 중에서는 Tanh 함수를 사용하는 것이 대부분 좋은 성능을 보였으나, TRPO와 같이 문제가 복잡해질수록 연산이 단순한 Clamping이 오히려 좋은 결과를 보여주었다. 리워드 함수에서 Transformer OverLoad에 과도한 과중치가 부과되어 리워드가 높을수록 유저 만족도를 무시하고, 충전에 소극적인 모습을 보여주었다.

5. 앞으로의 계획

5.1) 앞으로의 연구 방향

5.1.1) Action Sampling

PPO에 Action Sampling을 적용시켜 본 결과 기존 PPO 보다 확실하게 좋은 성능을 가져오는 것을 확인하였다. Action Sampling은 PPO에만 사용되는 것이 아닌, TRPO 등 많은 Actor Critic 기반의 강화학습 알고리즘에서 사용되는 것이기에, PPO 외의 알고리즘, 특히 대부분 좋은 성능을 유지하던 TRPO에 적용하여 더 좋은 결과를 만들고자 한다.

5.1.2) 유저 만족도

현제, Reward Function은 Transformer Overload에 상당히 소극적인 모습을 보인다. 이에, 유저 만족도가 떨어지더라도, Overload를 피하기위해 충전을 하지 않는 결과를 보여준다. 하지만, 이에 목표로 하는 SoC의 절반도 만족하지 못하는 결과를 주로 보여주며, 이는 V2G 사용자 참여에 악영향을 미친다. 이에, 새로운 Reward Function을 구상하여 유저 만족도를 높이면서, Overload는 방지하는 결과를 만들고자 한다.

5.1.3) 확장 가능성

현제, State Function은 Transformer와 EVSE의 결과를 정해둔 채로 이루어진다. 즉, Transformer나 EVSE의 개수가 달라지면 기존의 모델을 사용할 수 없게 된다. 이는 실제 상황에 적용하였을 때, 인프라가 늘어나면 다시 학습을 해야한다는 문제를 가져온다. 이에, State Function을 탐색하거나, Multi Agent를 사용한 강화학습을 통하여 인프라에 영향을 받지 않는 모델을 만들고자 한다.

5.2) 연구계획 및 구성원 역할 분담

5.2.1) 연구계획

구분	세부항목	9월				10월	
		1	2	3	4	1	2
설계	Reward Function 설계						
	State Function 설계						
구현	새로운 V2G 문제						
	Action Sampling TRPO						
실험	새로운 V2G 문제 성능						
	기존 문제와 결과 비교						
보고서	최종 보고서						

5.2.2) 구성원 역할 분담

학 번	이 름	구성원별 역할	
201924488	배레온	완료	- V2G 모델 수학적 분석 - V2G의 모델 설계 - 강화학습 알고리즘 개발 (DDPG, TRPO)
		예정	- Heuristics 알고리즘 개발 (AFAP, ALAP, RR) - 새로운 V2G 문제 정의 및 구현
202155517	김도균	완료	- 강화학습 알고리즘 개발 (A2C, PPO, SAC) - 논문 연구, 모델 설계 및 아키텍쳐 구상 - 샘플 코드 테스트 모니터링, 시뮬레이션 환경 구축
		예정	- MPC 알고리즘 개발 (eMPC V2G, OCMF V2G) - Action Sampling을 적용한 TRPO 개발 - Multi Agent RL 조사 및 개발

6. 참조

[1] “CFI 제주 2030, 탄소중립·에너지전환 선도”

[2] “무공해차 통합누리집”

[3] “EV2Gym: A Flexible V2G Simulator for EV Smart Charging Research and Benchmarking”

[4] “A Simulation Tool for V2G Enabled Demand Response Based on Model Predictive Control”

[5] “Asynchronous Methods for Deep Reinforcement Learning”

[6] “Continuous control with deep reinforcement learning”

[7] “Proximal Policy Optimization Algorithms”

[8] “Trust Region Policy Optimization”

[9] “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”