Ev ChargeEnv Documentation

This enviromnent simulates an EV Charging Park (aka EV Charging Plaza). This is a parking lot with multiple EV Charging stations (or Electic Vehicle Supply Equipment or EVSE). The name "EV Charging Park" (or park for short) is used emphasize the difference between a single EV Charging Station that can accommodate one Charging Session at a time and a set of them.

This environment runs on **discrete time**. The timestep is measured in minutes, and a common setting is $\Delta t = 15min$. [There is a reference that says that some countrie's regulations now focus on 15 min granularity instead of 1 hour ganularity, need to check].

State

The state takes inspiration from board games. In this case it's a single player game (PvE).

Board Game	EvCharge
Board	Charging park
Square	Parking spot
Piece	Car
Turn	Timestep
Move	*action*

Charging park

Defined by a single constant

- `max_cars`: The maximum number of cars in the Park. This is equivalent to the number of board

For now the size of the battery `C` will be the same for every car.

Therefore, the environment is a list of `max_cars` spots. Each spot can be occupied or vacant.

Car

The input is a DataFrame containing Cars (also called sessions or transactions).

The car iterator is $i \in \{0, 1, ..., max\_cars\}$ Each row will have:

→ Car constant attributes

- TransactionId (`idSess`): A unique identifier for that row (0 or -1 if spot is vacant)
- BatteryCapacity (`C`): The size of the battery (in Wh)
- TransactionStartTS (`t_arr`): The timestep in which the car arrives to the parking lot (in ts)
- `SOC_arr` (`soc_arr`): The State-Of-Charge with which the car arrives (from 0 to 1)
- TransactionStopTS (`t_dep`): Time of departure of the EV (in ts)

→ Car variable attributes

This variables are the ones that will be simulated (i.e. recalculated at every timestep)

- `soc_t`: State of charge at timestep t

The update rule is: $soc_t^i = (P_t^i \cdot \Delta t + soc_{t-1}^i)/C^i$

Although, by ignoring the constants, it can be simplified to: $soc_t^i = a_t^i + soc_{t-1}^i$

With a_t^i being the "normalized" power. In which case, there is no use for battery size, `C`.

→ Car auxiliary attributes

By simple arithmetic operations, the following features can be engineered. The value of them has not yet been measured.

- Sojurn time (`t_soj`): $t^i_{soj} = t^i_{dep} - t^i_{arr}$
- Energy at arrival (`E_arr`): $E^i_{arr} = soc^i_{arr} \cdot C^i$
- Energy at t (`E_t`): $E^i_t = soc^i_t \cdot C^i$
- Energy required (`E_req`): $E^i_{req} = C^i - E_t$
- Remaining time (`t_rem`): $t^i_{rem} = t^i_{dep} - t$

Observation space

In the simplest case, ignoring `C` and auxiliary features, the space will be:

`max_cars` * ({`idSess`, `t_arr`, `soc_arr`, `t_dep`, `soc_t`} + {Additional attributes})

In it's simplest form in RlGym terms:

`Box([0, 0, 0, 0, 0], [max_idSess, t_max, 1, t_max, 1], (max_cars, 5))`

I.e. `max_cars` squares with each piece having 5 attributes.

Additionally, the timestep t is a global variable.

Agent

The agent is the Virtual Power Plant (VPP) or Virtual Operating Reserve (VOR) that sets the power for each charging session.

Action space

The action will be $P_t = p^1_t, p^2_t, ..., p^{max\_cars}_t$ in Watts.

Or $A_t = a^1_t, a^2_t, ..., a^{max\_cars}_t$ in "normalized units".

In RlGym terms:

`Box(P_min, P_max, (max_cars, 1))`

Notes

In RlGym: `Box(lower_lim, upper_lim, shape)`