EV Planner local prediction

Created by Jia, Zhanfeng, last modified on Jul 09, 2024

› Revision History

Revision	Author	Date	Reviewer
e.g. "rev 0.1"	@Jia, Zhanfeng	08 Jul 2024	@Li, Mingzhou @Jiang, Ji @Sun, Xu (Curry)

1: Introduction & Background

To reduce EVAP http transmission cost and later Onboard EV trip plan, we need to research EVPlanner local prediction algorithm.

2: Requirements

Implement EVPlanner local prediction algorithm.

3: Acceptance Criteria

NULL

4: Design

4.1 Design brief:

EVPlanner uses EVAP's voting_table model for prediciton which is a table match algorithm.

Load EVAP's voting table locally and implement EVAP service's API from EV planner side.

4.1: EVAP service's API being used in EVPlanner.

1. Predict battery cost: /ev/predict/battery/batch
2. Predict distance: /ev/predict/distance/batch & /ev/predict/distance
3. Predict charging time: /ev/predict/charging/time/batch
4. Predict isochrone: /ev/predict/isochrone

4.3: Detail design:

4.3.1: Step-1 add new prediction model.

Add below red line prediciton model:

```
{
    linear,              // EVAP curve prediction
    voting,              // EVAP mach-learning prediction
    voting_table,      // EVAP table-match prediction
    .....
    local_voting       // EVPlanner local  prediction mode
}
```

4.3.2: Step-2 load EV vehicle details from EVAP.

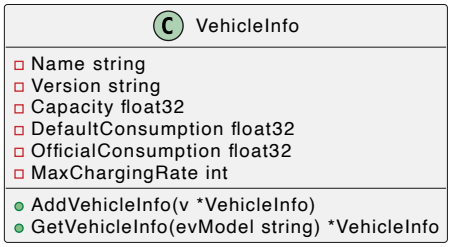
To support local prediciton, EVPlanner should knows EV vehicle detial's information.

EV Planner loads the EV vehicle details when startup.

EV Vehicle detials : EV vehicle details

EV vehicle profile class diagram desgin

Profile




```
add vehicle info {Name:byd_song-plus-ev Version: Capacity:87 DefaultConsumption:16.8 OfficialConsumption:15 MaxChargingRate:130}
add vehicle info {Name:byd_atto3 Version: Capacity:60.5 DefaultConsumption:15.6 OfficialConsumption:16 MaxChargingRate:80}
add vehicle info {Name:byd_han-ev Version: Capacity:85.4 DefaultConsumption:18.5 OfficialConsumption:18.5 MaxChargingRate:120}
add vehicle info {Name:byd_tang Version: Capacity:86.4 DefaultConsumption:23.8 OfficialConsumption:23.8 MaxChargingRate:120}
add vehicle info {Name:byd_seal Version: Capacity:82.5 DefaultConsumption:17 OfficialConsumption:16.6 MaxChargingRate:150}
add vehicle info {Name:byd_dolphin Version: Capacity:44.9 DefaultConsumption:16 OfficialConsumption:15.2 MaxChargingRate:60}
add vehicle info {Name:byd_seagull Version:eu-0xfb Capacity:43.2 DefaultConsumption:10.4 OfficialConsumption:10.4 MaxChargingRate:4
add vehicle info {Name:byd_tang Version:6125 Capacity:86.4 DefaultConsumption:23.8 OfficialConsumption:23.8 MaxChargingRate:110}
add vehicle info {Name:byd_dolphin Version:0x7f Capacity:44.9 DefaultConsumption:16 OfficialConsumption:15.2 MaxChargingRate:60}
add vehicle info {Name:byd_others Version: Capacity:60 DefaultConsumption:15 OfficialConsumption:15 MaxChargingRate:150}
add vehicle info {Name:byd_new-e5 Version: Capacity:52.8 DefaultConsumption:14.57 OfficialConsumption:14.57 MaxChargingRate:60}
add vehicle info {Name:byd_new-e6 Version: Capacity:55.4 DefaultConsumption:17.1 OfficialConsumption:17.1 MaxChargingRate:85}
add vehicle info {Name:byd_yuan-up Version: Capacity:45.12 DefaultConsumption:12 OfficialConsumption:12 MaxChargingRate:82}
add vehicle info {Name:byd_yuan-pro Version: Capacity:32 DefaultConsumption:12 OfficialConsumption:12 MaxChargingRate:45}
```

4.33 step-3 load predction talbe from EVAP

Currently, EVAP uses only one prediction table to predict all EV vehicle models, and EVPlanner only need to load this table at startup.

Prediciton talbe's format is csv.

```
quantity(kWh),A/C,ambient_temperature,ventilation,consumption(kWh/100km),avg_speed(km/h),pa
9.0,10,30,0,21,120,1,34.08
9.0,10,30,0,21,120,3,34.03
9.0,10,30,0,21,120,5,33.97
```

EVAP Prediction talbe description:

Parameter	Description
quantity(kwh)	battery cost, uint: kwh
A/C	air condition, unit: 0: turn off, 1: turn on, 10: heating on
ambient_temperature	ambient temperature, uint: centigrade
ventilation	ventilation, uint: 0:turn down, 1:low, 2:normal, 3:high.
consumption(kwh/100km)	battery consumption, uint kw/100km
avg_speed(km/h)	travel speed, uint(km/h)
passengers	passengers
trip_distance(km)	travle distance uint:(km)

Load predicotn table rules:

EV Planner loads the entire predciton table into memory at startup.

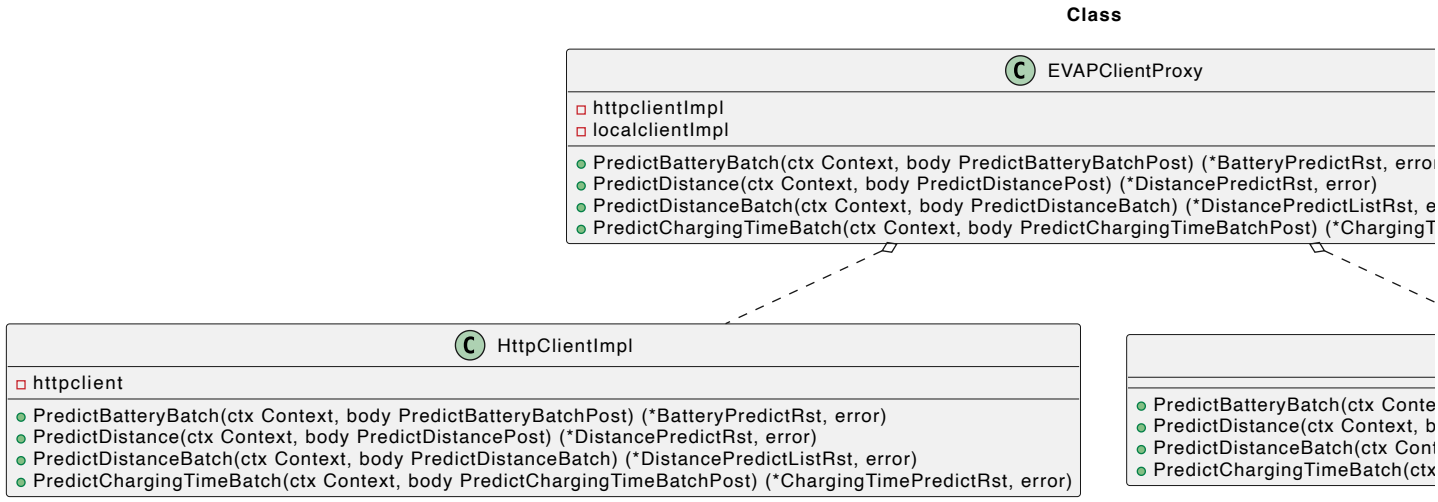
EV Planner will use AuxiliaryConsumption parameter to calulate batttery cost, so only the columns where 'A/C' and 'ventilation' is off will be loaded.

EVPlanner also ignores this column because the client cannot get the passengers.

According to the above loading rules, the result is as follows：

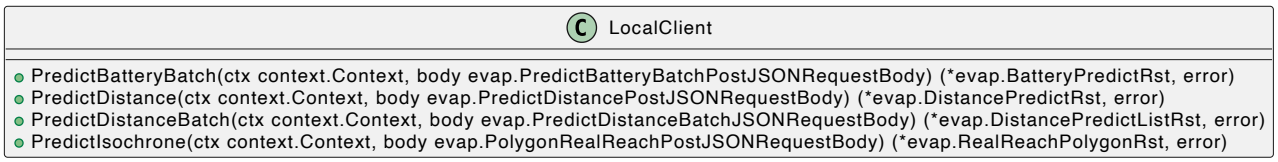
```
prediction_voting_table_model.go:237] total useable item cnt: 29992
prediction_voting_table_model.go:238] tripConsumptionMap item cnt: 184, item key(speed_consumption_temperature)
prediction_voting_table_model.go:239] quantity(kWh) min: 0.50, max: 145.00, item cnt: 163
prediction_voting_table_model.go:243] trip_distance(km) min: 0.54 max: 970.17, item cnt: 18603
prediction_voting_table_model.go:247] avg_speed(km/h) items: [10 20 40 60 80 100 120 140]
prediction_voting_table_model.go:248] consumption(kWh/100km) items: [15 17 18 19 20 21 22 23 24 25 26 28 30 32 34 36 38 40 4
prediction_voting_table_model.go:257] ambient_temperature: [24]
```

4.3.4: step-4 EVAP Client Proxy class diagram design

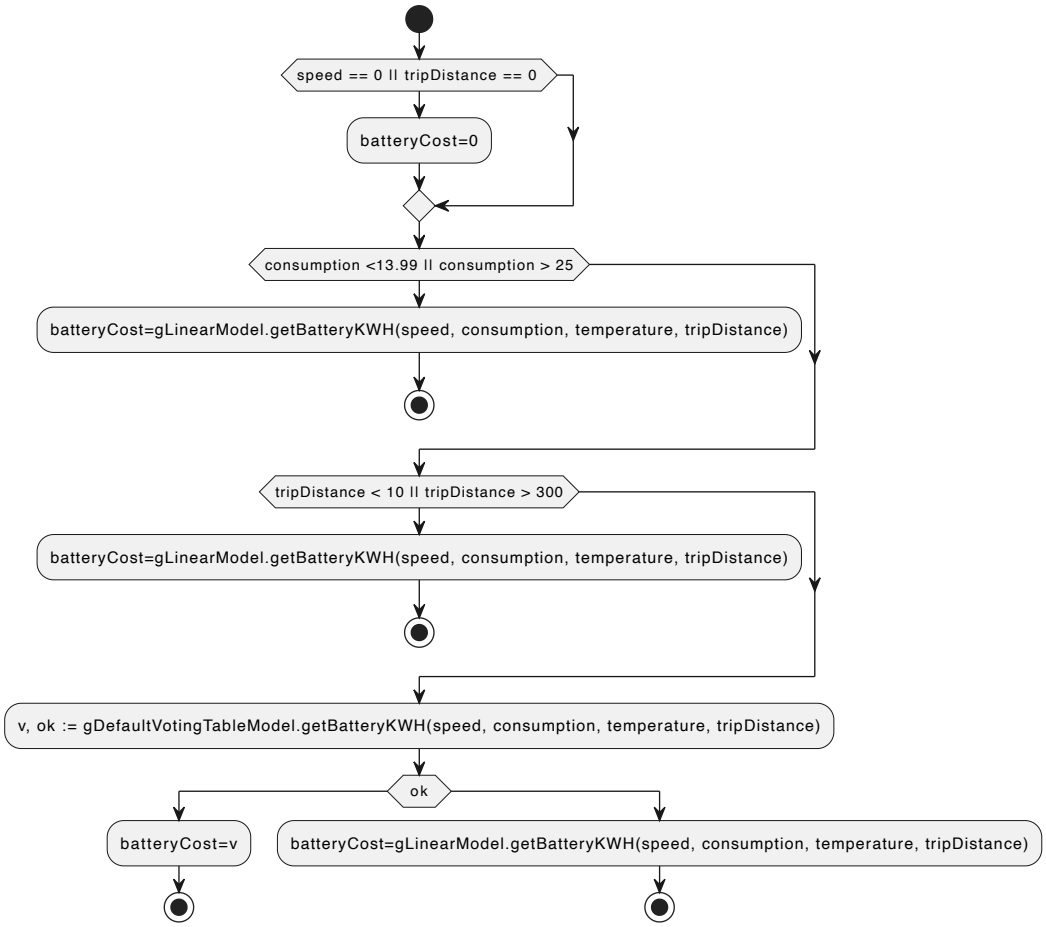


4.3.5: step-5 EV Planner local client design:

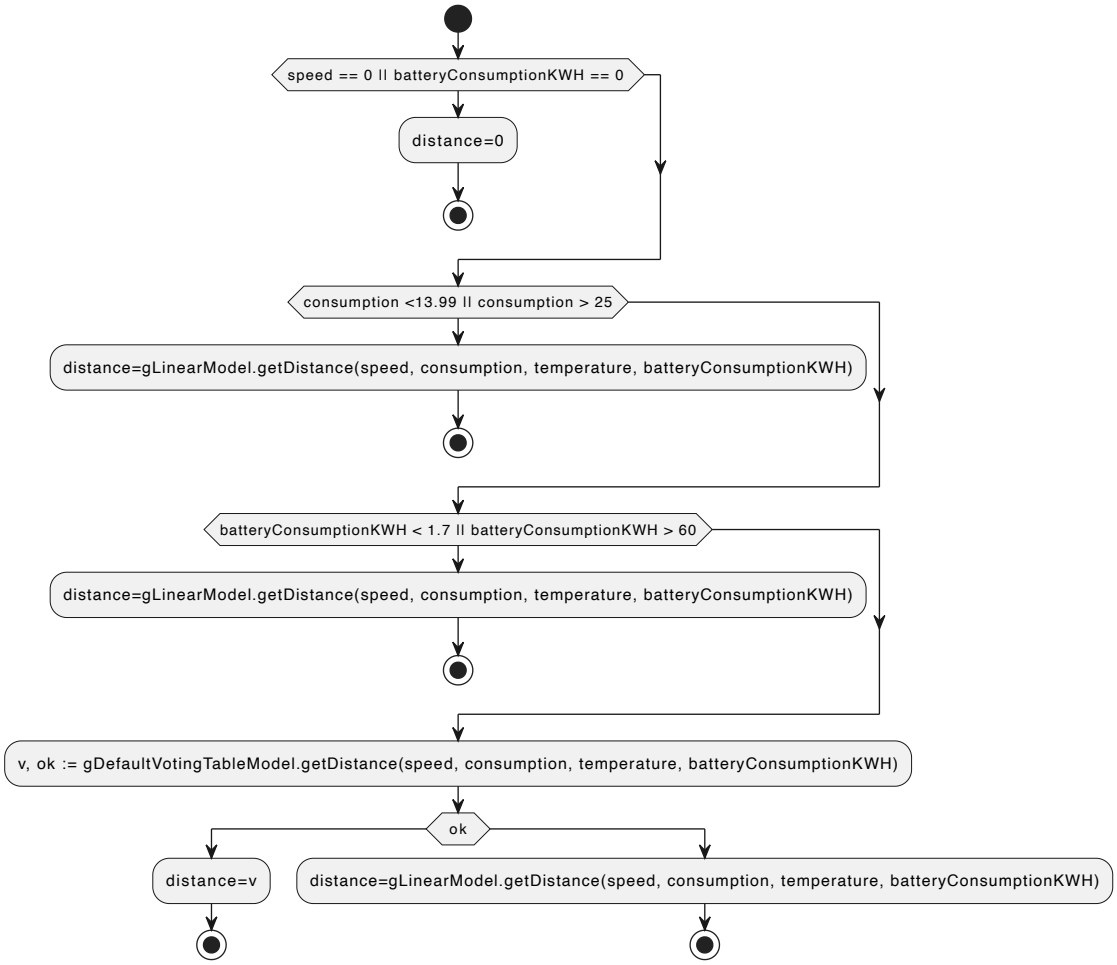
local client diagram desgin



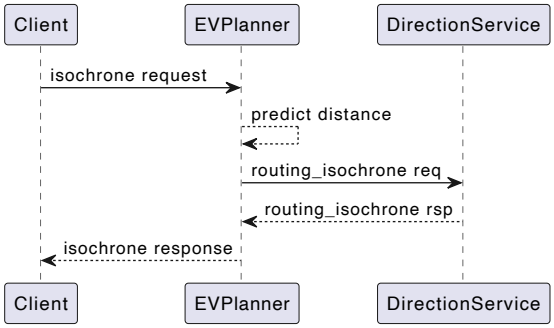
PredictBatteryBatch: /ev/predict/battery/batch



PredictDistance & PredictDistanceBatch: /ev/predict/distance & /ev/predict/distance/batch



Predict isochrone: /ev/predict/isochrone



4.3.6 Local Linear mode algorithm:

1. **Predict battery cost**

   API: /ev/predict/battery/batch

   ~~Formula: C=Pac*Δh+ + Pdec*Δh- + Pspeed*L +Paux*t~~

```
func (p *PredictionLinear) getDistance(speed int, consumption float32, temperature float32,
    coefficient := (1 + speedConsumptionCoefficient(speed))
    consumption = float32(coefficient * float64(consumption))
    return batteryConsumptionKWH * 100 / consumption
}
```

2. **Predcit distance**

   API: /ev/predict/distance & /ev/predict/distance/batch

```
func (p *PredictionLinear) getBatteryKWH(speed int, consumption float32, temperature float32,
    coefficient := (1 + speedConsumptionCoefficient(speed))
    consumption = float32(coefficient * float64(consumption))
    return tripDistance * consumption / 100
}
```

3. **Predict charging time**

   Not support

4.3.7:Local Voting talbe mode algorithm:

1. **Predict battery cost**

   API: /ev/predict/battery/batch

   Binary Search from prediction table.

2. **Predcit distance**

   API: /ev/predict/distance & /ev/predict/distance/batch

   Binary Search from prediction table.

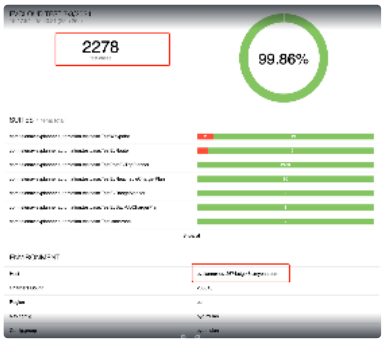
3. **Predict charging time**

   API: /ev/predict/charging/time/batch

   Not support

5:Test Report

Regression test report:

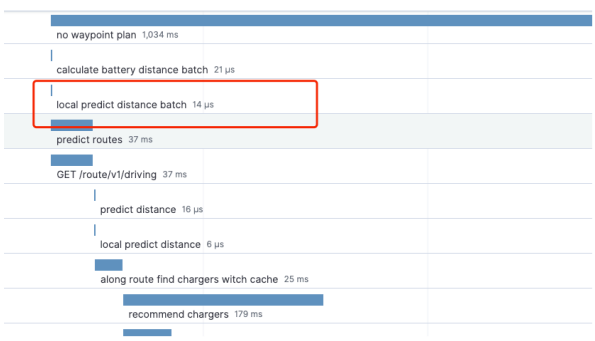


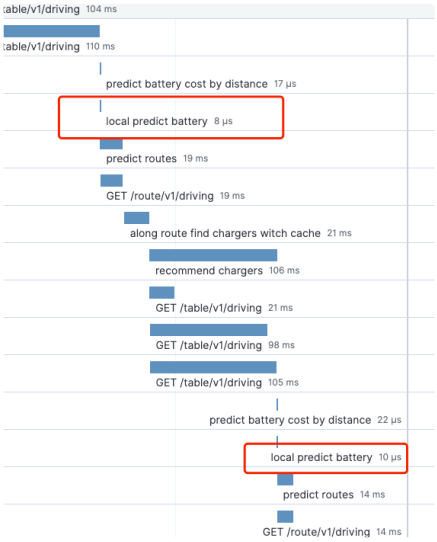
https://tactaws.telenav.com/report/EvCloudTest/here_eu/stg/Debug/here_eu_9865/v0.17.3/index.html

Performance test report:

distance	voting_table	local_table	improve
100KM	1.176s	0.952s	19%
500KM -1000KM	6.407s	4.253s	33%
1000KM-2000KM	8.239s	6.767s	17%
2000KM-3000KM	13.576s	11.629s	14%
3000KM-4000KM	18.069s	15.412s	14%

APM test report:





6.TO DO:

1: charging time table.

2: altitude battery compensatio.

3:auxiliary battery compensatio.

4: update prediction table dynamically .

5: prediciton table improvement. ~~DATAPLFORM-15105~~ - EVAP prediction table improvement **RESOLVED**

7.Reference

~~charging curve ref: https://www.gridserve.com/2023/02/17/what-is-an-electric-car-charging-curve/~~

~~HERE API ref: https://developer.here.com/documentation/routing-api/dev_guide/topics/use-cases/ev-routing.html~~

~~onboard voting predict ref: Onboard Predicting Feasibility~~

~~https://pod-point.com/guides/vehicles/tesla/2022/model-s~~

No labels