Technische Universität Berlin
Lecture: Fundamentals of Multi-Agent Technologies
Profesor: Stefan Fricke
Project: Assignment 2
Title: Cooperative Electric Vehicles in the Electric Grid
Date: January 6 , 2020

1 Introduction

The purpose of our implementation is to create an environment for successfully deploy a set of Cooperative Electric Vehicles in the Electric Grid.

We plan to integrate different classes of agents in the Cooperative Domain with multiple purposes and capabilities e.g. Cars and Charging Stations, that will be communicating with each other in order to request and provide different services.

The Programming language used through the project is Java and we will be relying in JADE (Java Agent DEvelopment Framework) as our Framework for the implementation of multi-agent systems.

In the beginning we were relying on JIAC (Java-based Intelligent Agent Componentware), that is another Framework for agent-based applications and services ,however there was no Communication Protocols available and creating a communication protocol is out of the scope of this assignment. For this reason we decided to switch to JADE.

The intended delivery will be a simulation in where the different agents cooperate with each other in order to fulfill with the highest satisfaction percentage the requirements of the cars to charge either fast or cheap and also manage the available slots on the different stations to evenly occupied the grid taking in consideration redirecting the flow of the oncoming charging requests to different stations either by offering a better price or forcing the superchargers in certain hot-spots to shut down to avoid an electric grid overload.

2 Methodology

There will be a set of Parameters that are important for creating a real-world behaviour inside our simulation [3]. So we will have two types of Chargers inside every station:

-Slow Charger: This will be the standard type of Charger and will constitute approximately the 70% of the offer on the Charging stations

-Fast Charger : This will be an Special Charger that will charge the car faster, but with a higher price per KW

2.1 Time vs. Money

The core idea of the project is to have two different behaviours of the clients(cars) when searching/requesting a service(charging station):

- Car Behaviour 1 : This car does not care about the price of the energy , it want to charge as quick as possible.
- Car Behaviour 2 : This car does not care about the time of charge , it wants to charge as cheap as possible.

Based on this priorities we have to find the best behaviour of the systems in order to have the highest amount of satisfied agents.

On top of this two premises different other factors need to be handled:

- The distance to the charging station can substantially affect the charging time.
- If a station is full the waiting cars can be redirected to nearby stations.
- In case of emergency (cars below 10% of charge) the priorities of the clients (charge asap) have to override the initial desire.
- In case of a Charging Station malfunction actions need to be taken to redirect the cars to nearby Stations.
- Schedule and book charging slots in advance to better utilize every Charging station time-frame, considering a 24hr service and knowing the users timetable (the owner of the car defines its weekly schedule so we know when the car is free for charging).

As noticed in Figure 1 there will be limited amounts of slow chargers available per station and for every step of the program the energy will be draining in the Cars depending on its status, therefore a management of the combined costs (distance, rate and priorities) is required , and this is the core of our behavioural algorithm.

It's important to notice as well that different stations will have different rates on the energy per KW/h , with this we are trying to emulate the actual situation of the petrol stations scheme, in where depending on the brand and position of the Station the price per liter varies[1].

The current price for Electric Vehicle Charging Stations is €0.10 per KW/h [2] so based on that information we will consider the cheapest slow charger on the Suburbs on that price , and every fast charger will cost the double of the price of the slow charger of the respective station and therefore calculated to charge twice as fast.

At the peak times the Cars that want to use the Fast charger may have to pay an extra fee every certain amount of time during this critical period (otherwise the the grid will be

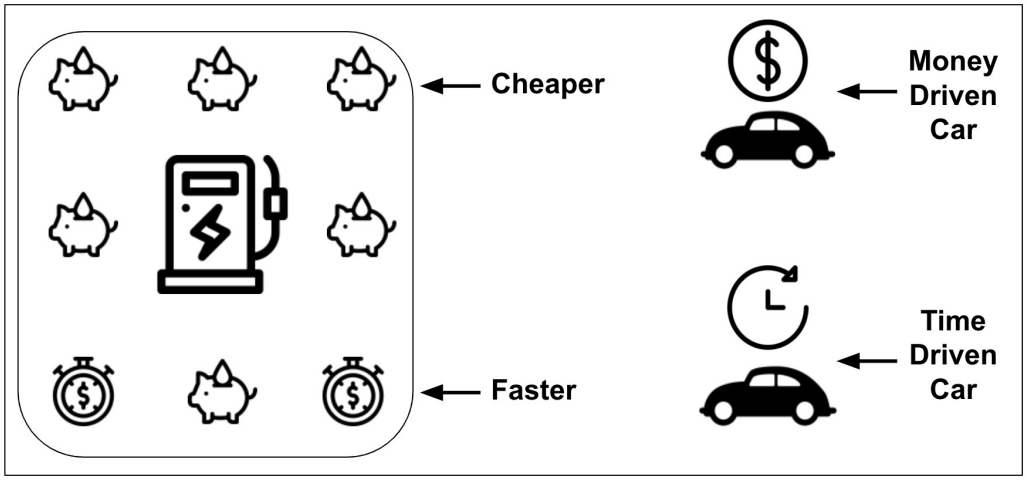

Figure 1: Representation of a Typical Charging Station (left) and of the two different Car Behaviours (right)

forever overloaded with cars charging at fast modes and the network can be damaged).

In our idea (Figure 2) the Stations located in the "Center" of the city will have a higher cost per KW/h (€0.15 per KW/h) related to the ones located in the "Suburbs"(€0.11 per KW/h) with that we will try to avoid congestion in the main transited areas and disperse the usage among to the adjacent less expensive stations, taking into consideration that the traveling distance implies a cost as well and balancing the time/cost to always pick the best possible solution.

3 Implementation

The implementation will be developed using Java+JADE and will consist in the following steps

The Java code will be created for managing every Agent in a separated way and in different classes as follows:

- For managing the Vehicles and it's behaviours inside the Grid we will create the "VehicleAgent"
- For the Charging Station Agent and it's management we will create the "ChargingStationAgent"
- For the Communication we will have special functions (included on JADE) that will be embedded in every Agent in order to be able to communicate on the Platform with all the other Agents according to the FIPA Standards

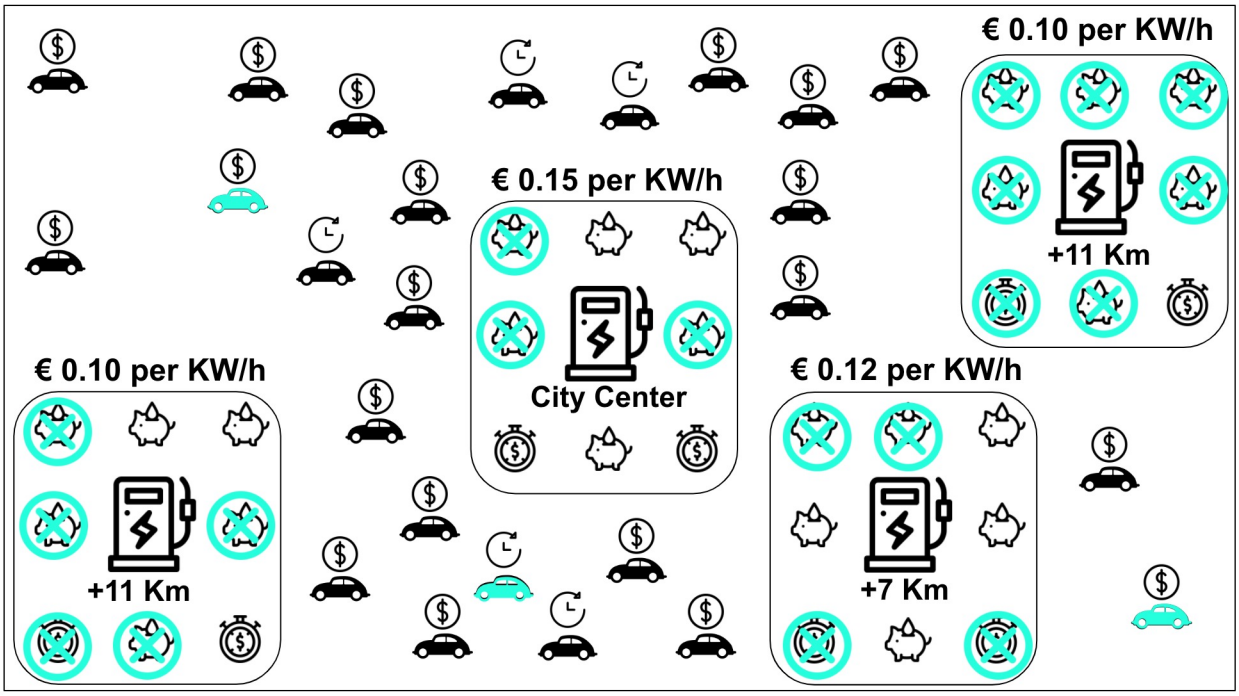

Figure 2: Multi-Agent System (Stations + Cars) negotiating in a real-style environment, cyan cars:emergency mode , cyan crosses:occupied chargers

- For the Simulation a set of files will be created in order to Manage a Graphical User Interface (GUI) that will help us to easily visualize the process and movement of every agent at every given step. That will allow us to easily understand if the algorithm is working properly and we will be able to see how every charging spot is being occupied by a Car and if all the cars get eventually charged whenever they need it.

3.1 Platform and Agents

3.1.1 JADE Requirements

For successfully initialize the JADE Framework a pair of resources are initialized [4] :

- The Agent Management System (AMS) is the agent who exerts supervisory control over access to and use of the Agent Platform. Each agent must register with an AMS in order to get a valid AID.
- The Directory Facilitator (DF) is the agent who provides the default yellow page service in the platform.

- The Message Transport System, also called Agent Communication Channel (ACC), is the software component controlling all the exchange of messages within the platform, including messages to/from remote platforms.

When a JADE platform is launched, the AMS and DF are immediately created. Furthermore the Messaging Service is always activated to allow message-based communication.

3.2 Yellow Pages

The Yellow Pages is a Service that contains the information required for publishing and searching for services through a variety of method calls. This service is provided by the Directory Facilitator Agent and is available to all the agents.

The two main goals of the Yellow Pages are:

Publishing Services:

An agent wishing to publish one or more services must provide the DF with a description that includes its own AID and the list of provided services.In our case the Charging Stations will be publishing to the Yellow Pages relevant parameters and information such as the number of available slots , the price of the energy at this given step

Searching for Services:

An agent wishing to search for services must provide the DF with a template description. The result of the search is a list of all the descriptions that match the provided template. In our case the Cars are going to be searching first the nearby Charging Stations with spots available and will retrieve the cost and schedule and then request the preferred service among all the matching options.

3.3 Negotiation(ContractNet)

For the Negotiation Part between the Agents we will be relying in the Contract Net Protocol [5] . The Car will Act as Manager and on the time when a charge is needed will call the nearest stations(contractors) for proposals , after that the proposals are going to arrive to the car and then the car will be able to either Accept or Deny the oncoming proposals and send back the response to finally receive a confirmation from the Station where the proposal was accepted and finish the contracting task. In our case the proposals are numbers of two kinds. Negative when the station is not able to satisfy the vehicle agent and a ratio which is a combination result of distance and agent's goal.

3.4 Simulation and Results

For the Simulation a Simple GUI was chosen in order to create a grid that can take 4 different states associated with different colors (Fig. 3):

- White Cells: All the non occupied space
- Blue Cells: Charging Ports inside the Charging Station
- Red Cells: Cars , able to move across the Black and Blue Cells
- Grey Cells: All the Streets, where the Cars can drive on

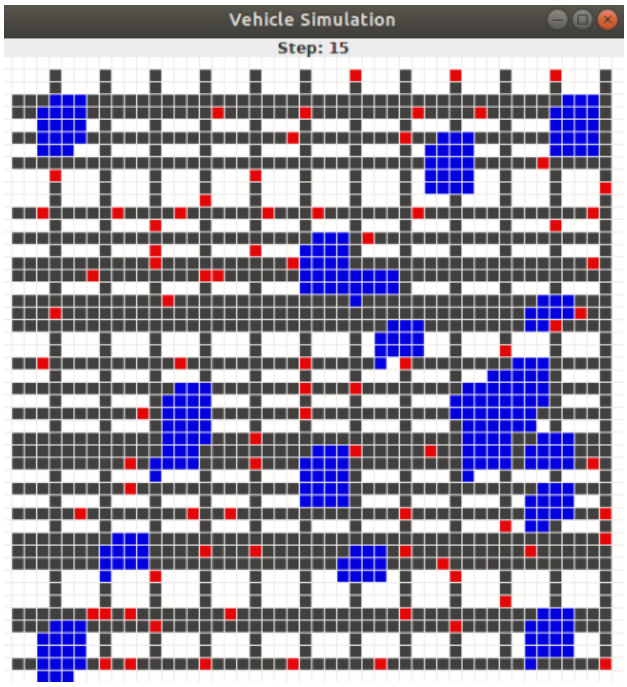

Figure 3: 2D Simulation grid, showing the vehicle agents and charging stations.

The simulation show the travel of the Car from going around the field(always over the black cells) to start negotiating with the adjacent Charging Stations whenever a charge is needed and eventually accepting an offer and entering the designated blue area where it will stop while charging and the abandoning and continue discharging until eventually start a contract with the next adjacent Charging Station.

4 Future Ideas

Charging Event linked to Chargers

In a future and more complex oriented implementation the Charging Events are going to be linked to Chargers objects and not to the Station Agents. At the time being, the Charging event contains all the information about a charging, they are placed in a list of charging events in station agent object. Considering 4 fast chargers and 4 slow chargers (8 in total) in a station, 8 different lists are going to be crated for every kind of charger. This is going to allow better management of the charging station entries in yellow pages and faster and safer communication in negotiations.

Intelligent Charging Station Scheduling

At the moment the booking list of the charging stations have many empty spaces, where no cars are charging. One way to optimize the charging times would be to let cars not to fully charge.

Map Agent

A Map Agent could help to better compute distances between the Vehicle Agents and the Charging Stations. Now we calculate the distance with the Breadth-first Search Algorithm and have to explicitly call the function. With a Map Agent we could more compute the nearest charging station at each time step.

Time Agent

Allow users to start and stop a simulation case and set the simulation step and time of the simulation case, keep synchronized the time of the steps and actual running time. Also, would be useful to lock execution threads in crucial part of simulation like negations and charging events.

Power Grid Agent

The power grid agent may make different sales price according to its design objective. For example, the power grid agent may make proper sales price to reduce the peak-valley difference. Also the charging station and vehicle agents report their provided, consumed or existence power to the power grid agent which keeps the whole track of them. Also this agent is responsible for avoiding the overload, announcing to all agent problems in charging stations or vehicles and play the role of coordinator.

Adaptive Charging Agents

One Idea for further implementation is to have a third Car Agent that is able to now establish his desired type of charge depending on a balance based on a given time frame. e.g if we know that the user is not going to be occupying the car in three hours and the car will fully charge in four hours on slow mode then we can go one hour into fast charge and the remaining time into slow charge to fully charge the car at the end of that time and be able to make the best cost/benefit balance for the given time.

Ranges of protocols

By implementing different protocols and not just contract net, better overall coverage and higher levels of efficiency could be achieved within the grid. Having the likes of a blackboard implemented could allow for some of the strain being taken of the agents to have to go through the entire process of proposing offers to the vehicles. An instance of this could be when the grid is off-peak and there are multitudes of spaces available and thus the Contract Net would be redundant.

5 Conclusions

A Multi-Agent System is a complex development that require interactions between different entities in order to propose , negotiate , manage and implement diverse behaviours , services and decisions.

In our project we manage to successfully communicate the cars to the adjacent stations while requiring a charge, negotiate a price and/or estimated charging time with all of them and then decide and approach the chosen one.

This projects is a summary of the deployment of a good example of Cooperative Multi-agent System and requires a good background and knowledge of the implemented Protocols like ContractNet and FIPA regulations and a good basis and programming skills for developing the interactions between the agents.

6 References

[1] `https://gaspricesexplained.com/#/?section=gasoline-diesel-and-crude-oil-prices`

[2] `https://afdc.energy.gov/fuels/electricity_charging_home.html`

[3] `http://nationale-plattform-elektromobilitaet.de/fileadmin/user_upload/Redaktion/AG3_Statusbericht_LIS_2015_engl_klein_bf.pdf`

[4] `https://jade.tilab.com/doc/programmersguide.pdf`

[5] `https://en.wikipedia.org/wiki/Contract_Net_Protocol`