Using Constraint Programming to Optimize the Recharge Operations of a Fleet of Electric Cabs

Kim Rioux-Paradis[1] and Claude-Guy Quimper[1]

Université Laval, Quebec QC, CAN
kim.rioux-paradis.1@ulaval.ca
claude-guy.quimper@ift.ulaval.ca

Abstract. We present a new constraint programming model and a heuristic approach for optimizing the charging operations of an electric taxi fleet at Téo Taxi. The goal is to maximize the charge of the cabs during high-demand periods. We use simulation to compare the solutions provided by the constraint solver and the heuristic method.

Keywords: Constraint Programming · Cabs fleet · Modeling · Simulation

1 Introduction

The cab industry faces multiple optimization problems. For instance, minizing the time to pick up a customers in order to maximizes the number of rides accomplished during a work shift. Software can help to take the decisions that provide a good dispatch. But the problem remains difficult as pickup addresses are not known in advance. More recently, new optimization problems arrised with electric cars. Indeed, electric cab companies such as Téo Taxi need to plan their charging operations in order to get the cab fleet ready for rush hours. These charging operations need to be integrated to the dispatch system. The operator can either send a cab to pick up a customer or go charging to be ready for the next rides.

The goal of this paper is to present a new optimization model for the charging plan of the vehicles. The next section will describe the problematic in more details. Then, we will review the literature about this problem. After, we will present the policies we propose. We will show how we want to evaluate them and what are the data we need. Finally, we will present future work and a conclusion.

2 Problem description

Téo Taxi owns a fleet of 100% electric cabs. The drivers of Téo Taxi are employees. They have fixed eight-hour schedules and a fixed salary. Unfortunately, cabs do not have enough battery autonomy for an eight-hour shift. They need to charge the vehicles once to six times per shift. At Téo Taxi, there is always a cab plugged on a charging station. When the driver's cab battery is low, the

driver goes to a charging station and switches cabs. S/he plugs the cab with a low battery and leaves with the cab with a recharged one. This is called a *permutation*.

We aim to plan the charging operations of their fleet. The goal is to have a model providing a schedule of permutations based on demand. Since the permutations happen often, we need to do this affectation in between the cabs-customer affectations. The objective of the schedule is to have a fleet of vehicles with enough remaining battery capacity for the high demand period and to leave sufficient time between two permutations to allow the batteries to fully recharge.

3 Literature review

The literature mostly covers the problem of assigning cabs to customers. When a customer asks for a vehicle, one needs to choose which cab in the fleet is going to be assigned to the customer. Each driver wants to maximize his/her own profit. The profit of a driver is the gain for a ride minus the expenses. For example, a long ride is more lucrative than a small, except if the customer is initially too far from the driver. With electric cabs, one can consider the charging stations as customers ordering a cab. These dummy rides have no profit, but have a very high penalty if they are not fulfilled since an uncharged cab needs to be towed and therefore is temporarily withdrawn from the fleet.

In a situation where we treat the customers first-in first-out, we can assign the nearest cab available. This technique is not efficient if the demand is high. A cab really far from a customer will be assigned to this customer if it is the only one available. However, if a closer cab gets available one minutes later, the assignment will not change even if the new cab is a better fit. Hence, the first-in/first-out strategy leads to an almost random assignment [6]. A driver might also drive more to reach the customer than to perform the ride, which leads to a loss of money.

To increase the driver's profit, we can use a matching. We pair drivers and customers. Each driver wants to maximize his/her profit and each customer wants to minimize the waiting time. We want pairs such that no pair of drivers would both agree to exchange their customers. [1] This is called a stable matching.

To improve the quality of the matching, a strategy consists in not immediately assigning cabs to customers [3]. Hence, all customers ordering a cab between a period starting at time a and finishing at time b are going to be assigned at time b using a stable matching. Note that the duration $b - a$ has to be smaller than the time a customer is ready to wait. Otherwise, she/he will cancel the ride before being assigned.

To decrease the computation time, Maciejewski et al. [6] limit the list of candidates for a cient to the k-nearest cabs. To find a perfect matching between the cabs and the customers, one can use the Gale-Shapley algorithm [2]. This algorithm finds a stable matching between two lists of n elements. Kummel et al. [3] adapt it to a situation where the length of each list is different.

The emergence of electric cars allows cab companies to decrease the gas cost. However, this brings new challenges because of the limited battery capacity of the vehicles compared to a gas vehicle. Moreover, the recharge time of an electric vehicle is much longer than refuel time. The assignment algorithms of customer-cabs have to be rethought, because a cab might not have enough remaining battery capacity to complete a ride that should have been assigned to it. Consequently, one has to include the charging planning in the assignment algorithm.

La Rocca and Cordeau [4] study the assignment problem for electric cabs. They reuse previous assignment algorithms by estimating the consumption of a ride and using only the cabs with sufficient charge in the selection. One heuristic they developed includes the availability of the charging station. If the number of available charging stations is high, the cab with less remaining battery capacity is assigned to the customer since the cab can go to the charging station after the ride. Otherwise, the cab with the most remaining battery capacity is assigned to the customer [5].

Zhou et al. [7] use game theory to decide at which station a cab charges. More precisely, they use Stackelberg games [7] where there are leaders and followers. The leaders tell what they are going to do and the followers chose the best response. The driver asks for a price to the charging stations when his/her cab needs to be charged. The charging station responds with a price based on the demand, their availability, the electricity price, and the distance between the charging station and the driver. The Nash equilibrium reveals which charging station the driver should go.

La Rocca et al. [4] developed a heuristic based on the demand. It has two thresholds, one when the utilisation of the cab is high Θ^{high} and one when it is low Θ^{low} where $\Theta^{\text{high}} > \Theta^{\text{low}}$ When the battery charge rate of a taxi is smaller than the threshold of the current utilisation, the taxi is sent for a permutation. The thresholds are therefore variable and depend of the utilisation of the cab.

Before adopting an assignment policy, one can evaluate it using a simulator [4]. The simulation can compute performance metrics to compare which policy is the best. A discrete event simulator is the strategy the most used. In such a simulator, each event happens at a discrete time and changes the state of the simulation. In many simulators, the charging plans are recomputed after every event [4, 5]. Often, the distances between drivers, customers, and charging stations are approximated using the Euclidian distance between two positions [4, 3].

Depending on the goal of the policy, different metrics can be used to evaluate the quality of the planning. For instance, Zhou et al. [7] evaluate how the electric vehicles are evenly allocated to the charging stations in order to avoid queues on some charging stations while others are idle. La Rocca and Cordeau measure the idle time of each cab, the number of completed rides, and the waiting time at the charging stations. In the case where all the cabs are owned by the same company and drivers get a fixed hourly salary, the idle time of one specific driver

does not matter. Hence, one can use the hourly income or the waiting time of the customers as a metric [4]

4 Policies

A policy is the decision process one follows to obtain a solution. In our case, a policy decides when and to which charging station a cab needs to go. A policy takes as input the state of the fleet, i.e. the charging status of the cabs' battery, the schedule of the drivers, the state of the cabs (in service, charging, in transit for charging, idle), and the demand (the expected number of rides in the coming horizon). For the first phase of this research, we do not consider the geographic location of neither the cabs nor the charging stations. As output, the policy returns the time at which each cab needs to transit for a permutation at the charging station as well as which charging station the cab is sent to.

We use different metrics to evaluate the efficiency of a policy: the number of towings (when cabs cannot reach a charging station), the average waiting time of the customers, and the number of cancelled rides.

Some policies are very fast to apply as they are simple rules. They can be used to replan the charging schedule on the fly, whenever a change of situation occurs (even minor changes). Other policies, like those based on optimisation techniques, can take some time to compute and therefore are applied periodically, say, every hour.

4.1 Heuristic

One heuristic we want to try is similar to the one developed by La Rocca et al. [4]. Our heuristic reacts to the demand. We define the demand as the number of rides in an hour for a day in the week. This information can be directly estimated from historical data. When the demand is high, we want to do as few permutations as possible because the driver needs to achieve as many rides as possible. Therefore, only cabs with less than 5% of remaining battery capacity are allowed to permute. Otherwise, when the demand is low, we allow more permutations in order to restore the fleet before the next high demand period. In such case, cabs with 15% of remaining battery capacity or less are allowed to permute.

4.2 Constraint Optimisation Problem

We design a policy that takes decisions based on the result of an optimization. We model the problem as a constraint optimization problem, whose solution is a schedule of the permutations that maximizes the average charge of the fleet.

The model takes as input the following parameters. There are n_d drivers and n_b charging stations. The schedule has a planning horizon of $h = 1$ hour. The time for charging a car is R but the driver only needs b units of time to permute the uncharged car with the charged one at the station. The initial charge of

driver i's cab is a_i and the maximum charge a cab can have is λ. The decharging rate of a cab is approximated with ρ.

The complete optimization model is given below. The decision variable U_i is a Booleean variable that is true if the driver i goes to the charging station in the coming hour. Should driver i permute his/her car, the decision variable T_i tells at what time driver i is going to permute. Finaly, the variable A_i is driver i's cab remaining battery capacity at the end of the one-hour schedule.

There are only three constraints in our model. Constraint (1m) assigns the final remaining battery capacity of the driver's cab. If the cab goes to a charging station, the final remaining battery capacity is the maximum battery capacity minus the discharged occuring between the end of the permutation and the horizon. If the cab does not go to a charging station, then the final remaining battery capacity is the initial remaining battery capacity minus the discharge during the horizon. In either case, the discharge is approximated with a linear function.

We use a CUMULATIVE constraint (1n) to ensure that cars are given a minimum of R units of time to recharge. This spreads out over time the arrival of cars at the charging stations. A car i uses U_i units of the resource. Therefore, cars that do not go to the station ($U_i = 0$) are ignored by the constraint.

Finaly, the third constraint (1o) breaks a symmetry in the model. If a car does not need to go to the station, the charging time is set to a constant.

The objective (1l) is to maximize the remaining battery capacity at the end of the hour.

The work being in progress, we did not decide yet which search strategy and branching heuristic should be used. We aim at implementing our model with MiniZinc and use the solver Chuffed.

5 Simulation

Before using the policies in the real planning operations, we are going to evaluate them with a simulation. The simulator is meant to help us convince the managers to use our policies and to determine which policy is the best to implement in real life. Our simulator is a discrete event simulator that we build from scratch in Python. Events include "new ride request", "start ride", "end ride", "permutation", and "towing".

5.1 Policies evaluation

We evaluate both policies: the one based on the heuristic and the one based on the constraint optimisation problem. Both policies are evaluated as they would be used in practice. For the heuristic policy, at the end of each event, we check if a permutation is needed or not. It so, the cab is immediately sent, on the fly, to a charging station.

The constraint optimization takes more time to evaluate than the heuristics. Therefore, we cannot solve it after each event like for the heuristic. We want

$$\textbf{Parameters: } n_d : \text{Number of drivers} \tag{1a}$$
$$n_b : \text{Number of charging stations} \tag{1b}$$
$$\lambda : \text{Maximum battery capacity of a cab} \tag{1c}$$
$$h : \text{Planning horizon} \tag{1d}$$
$$R : \text{Time for a cab to recharge} \tag{1e}$$
$$b : \text{Time for a driver to permute cars} \tag{1f}$$
$$\rho : \text{Decharging rate} \tag{1g}$$
$$a_i : \text{Initial remaining battery capacity of the driver } i\text{'s cab} \tag{1h}$$
$$\textbf{Variables: } T_i : \text{Permutation time of driver } i \tag{1i}$$
$$U_i : 1 \text{ if the car goes to permutation and 0 otherwise} \tag{1j}$$
$$A_i : \text{Final remaining battery capacity of the i's driver's cab} \tag{1k}$$
$$\textbf{Maximize: } \sum_{i=1}^{n_d} A_i \tag{1l}$$
$$\textbf{Under constraints: } A_i = \begin{cases} a_i - h \times \rho & \text{if } U_i = 0 \\ \lambda - \rho \times (h - (T_i + b)) & \text{if } U_i = 1 \end{cases} \forall\, i \in \{1, \ldots, n_d\} \tag{1m}$$
$$\textsc{Cumulative}(T, R, U, n_b) \tag{1n}$$
$$U_i = 0 \Rightarrow T_i = h \tag{1o}$$
$$\text{dom}(T_i) = [0, h) \tag{1p}$$
$$\text{dom}(U_i) = \{0, 1\} \tag{1q}$$
$$\text{dom}(A_i) = [5\%, 100\%) \tag{1r}$$

Model 1: CP model

to plan the permutations once every hour. This means, every hour, we solve the model and send the cabs for permutation according to the last computed schedule.

As the simulation evolves, statistics are gathered to evaluate the performance of the policies. We keep track of the average charging rate of the cabs, the average percentage of cabs with a charge higher than a given threshold, the number of cabs towed, and, most importantly, the number of rides.

5.2 Data analysis

The simulation instances are created by analyzing the data collected by Téo Taxi. The number of drivers and their respective schedules are drawn from the historical data. We do not compute statistical distribution for the schedules. We simply use one that actually occured in the past.

We derive a satistical distribution of the rides as follows. From the historical data, we analyze the number of rides by a granularity of one hour for each day of the week. We compute the mean number of rides for each hour to find the parameter of the Poisson process of the rides. Then, we generate new rides for each hour with that Poisson process.

To determine the length (in kilometres) of a ride, we analyze the mean and standard deviation of the length of the rides in the historical data and deduce the normal distribution. For each ride, we randomly generate a length according to this distribution.

6 Conclusion

The next step of this project is to complete the implementation of the constraint model and integrate it to the simulator. Then, we want to compute the efficency metrics for each policy and to compare them. We also want to compare with an off-line solution. This solution would be obtained from an optimization model that knows the future and always takes the best decisions. This policy is an oracle that knows exactly how the cab is going to discharge. This off-line solution represents the best one can do. We could compare how far we are from the off-line solution.

We also want to try a model where the horizon is 24 hours instead of one. In this case, we cannot assume that only one permutation occur during the planning horizon. We also have to take as input the schedule of the drivers and the cab that are assigned to this driver.

A better policy for managing the permuations will improve the efficiency of the taxi fleet. More cabs will be available during high-demand periods. More rides will be accomplished and towing will be avoided.

We believe that managing the permutations of an electric taxi fleet can also apply to a fleet of cargo trucks and that this research goes beyond the taxi industry.

References

1. Bai, R., Li, J., Atkin, J.A., Kendall, G.: A novel approach to independent taxi scheduling problem based on stable matching. Journal of the Operational Research Society **65**(10), 1501–1510 (2014)
2. Gale, D., Shapley, L.: College admissions and the stability of marriage pp. 9–15 (1969)
3. Kümmel, M., Busch, F., Wang, D.Z.: Taxi dispatching and stable marriage. Procedia Computer Science **83**, 163–170 (2016)
4. La Rocca, C.R., Cordeau, J.F.: Heuristics for electric taxi fleet management at teo taxi. INFOR: Information Systems and Operational Research pp. 1–25 (2019)
5. Lu, J.L., Yeh, M.Y., Hsu, Y.C., Yang, S.N., Gan, C.H., Chen, M.S.: Operating electric taxi fleets: A new dispatching strategy with charging plans. In: 2012 IEEE International Electric Vehicle Conference. pp. 1–8. IEEE (2012)

6. Maciejewski, M., Bischoff, J., Nagel, K.: An assignment-based approach to efficient real-time city-scale taxi dispatching. IEEE Intelligent Systems **31**(1), 68–77 (2016)
7. Zhou, H., Liu, C., Yang, B., Guan, X.: Optimal dispatch of electric taxis and price making of charging stations using stackelberg game. In: IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society. pp. 004929–004934. IEEE (2015)