Sorbonne Université
Master ANDROIDE

# Optimisation distribuée pour la recharge de véhicules électriques

Rapport de stage

Réalisé par :

**Xinyu HUANG**

Encadré par :

Laurent Pfeiffer, INRIA et L2S, CentraleSupélec
Guilhem Dupuis, EDF
Nadia Oudjane, EDF
Cheng Wan, EDF

Référent :
Bruno Escoffier, LIP6, Sorbonne Université

September 29, 2023

# Contents

# Chapter 1

# Problem formulation

This internship concerning a research project financed by the Programme Gaspard Monge pour l'Optimisation (PGMO), is co-supervised on one side by Laurent Pfeiffer, researcher at L2S, INRIA-Saclay, on the other side by Nadia Oudjane, Cheng Wan, and Guilhem Dupuis, researchers at EDF Lab Saclay.

**Keywords:**
Frank&Wolfe algorithm, smart charging problem, convex optimization.

**Objectives:**
Conduct a comprehensive study and implementation of the stochastic Frank&Wolfe algorithm (SFW) as introduced in the article by Liu et al.[1]. A detailed exposition of the SFW algorithm is provided in Section 2.2. Subsequently, explore the potential for enhancing the algorithm's performance by proposing and implementing several variants of the SFW algorithm or by reformulating the problem in novel ways to achieve a more robust solution, as detailed in Sections 3 and 4. Assess the efficiency of the SFW algorithm and its variants across diverse instances of a vehicle charging problem, in particular with varying values of the number of vehicles $N$ (Section 5). Finally, draw conclusions based on the observed results for all algorithms under consideration, highlighting any inherent limitations in existing approaches, and recommend future research directions and questions for further investigation for the vehicle charging problem (Section 6).

## 1.1 Context and detail of the problem studied

With the ongoing economic development, an increasing emphasis is placed on addressing ecological concerns and minimizing economic losses, particularly within the industrial sector. Our focus revolves around adapting electricity production to ensure a delicate balance between supply and demand. Within this context, we consider a charging station capable of generating electricity at each time step. This electricity serves not only a fleet of electric vehicles but can also be used to satisfy external demands. Consequently, our primary objective is to devise an optimal charging solution that efficiently manages both electricity production and distribution, aiming to minimize losses.

This problem can be formulated as a large scale, non-convex optimisation problem with a specific "aggregative" structure, which means that the objective function can be expressed as a function of an aggregate, the summation of electricity consummation of all the vehicles.

## 1.2 Aggregative problem

In order to facilitate the description of the smart charging problem of interest, we first formulate a general aggregative optimization problem, as defined in [1]:

$$\inf_{x\in\mathcal{X}} J(x) = f\Big(\frac{1}{N}\sum_{i=1}^{N} g_i(x_i)\Big) + \frac{1}{N}\sum_{i=1}^{N} h_i(x_i) \qquad (\mathcal{P})$$

where $x = (x_1, \ldots, x_N) \in \mathcal{X} = \prod_{i=1}^{N} \mathcal{X}_i$ and where the following data is given:

- $g_i : \mathcal{X}_i \to \mathbb{R}^n, i = 1, \ldots, N$
- $h_i : \mathcal{X}_i \to \mathbb{R}, i = 1, \ldots, N$
- $f : \mathbb{R}^n \to \mathbb{R}$.

The problem can be interpreted as a multi-agent model in which

- $N$ is the number of agents
- $\mathcal{X}_i$is the decision set of agent $i$
- $h_i(x_i)$ is individual cost function of agent $i$
- $g_i(x_i)$ is contribution of agent $i$ to a common good
- $\frac{1}{N}\sum_{i=1}^{N} g_i(x_i)$ describes a common good, referred to as aggregate
- $f$: a social cost associated with the aggregate.

## 1.3 Original problem: vehicles charging problem

In the context of the internship, we have focused on a specific instance of problem ($\mathcal{P}$) modeling a vehicle charging problem. Specifically, the variable $x_i$ describes the charging plan for the $i$-th electric vehicle:

$$x_i = (s_i^{t'}, \hat{s}_i^{t'}, c_i^t, \hat{c}_i^t, d_i^t, \hat{d}_i^t, u_i^t, \hat{u}_i^t, v_i^t, \hat{v}_i^t), \quad t \in \{1, \cdots, T\},\ t' \in \{0, \cdots, T\}.$$

**Variables:**

| Variables (baseline) | Variables (service) | Description | Range | Time |
|---|---|---|---|---|
| $s_i^t$ | $\hat{s}_i^t$ | The state of charge of the electric vehicle. | $[0, s_i^{max}]$ | $\{0, \cdots, T\}$ |
| $c_i^t$ | $\hat{c}_i^t$ | The charging power of the electric vehicle. | $\{0\} \cup [c_i^{min}, c_i^{max}]$ | $\{1, \cdots, T\}$ |
| $d_i^t$ | $\hat{d}_i^t$ | The discharging power of the electric vehicle. | $\{0\} \cup [d_i^{min}, d_i^{max}]$ | $\{1, \cdots, T\}$ |
| $u_i^t$ | $\hat{u}_i^t$ | The decision variable characterizes whether the vehicle is being charged or not. | {0,1} | $\{1, \cdots, T\}$ |
| $v_i^t$ | $\hat{v}_i^t$ | The decision variable characterizes whether the vehicle is being discharged or not. | {0,1} | $\{1, \cdots, T\}$ |

Table 1.1: Variables introduced in the origin problem.

**Parameters:** We classify parameters into three categories : parameters introduced in $\mathcal{X}_i$, introduced in the function $h_i$ and parameters introduced in the function $f$:

- Parameters introduced in the $\mathcal{X}_i$, describing the feasible set of vehicle $i$:
  - $c_i^{min}$ and $c_i^{max}$ : The minimal and maximal charging powers.
  - $d_i^{min}$ and $d_i^{max}$ : The minimal and maximal discharging powers.
  - $s_{i,init}$ : The initial state of charge.
  - $s_i^{final}$ : The targeted state of charge at the final time step $T$
  - $s_i^{t,min} = \max(s_i^{min}, s_i^{final} - (T-t)c_i^{max})$ : The minimal state of charge for the at time $t$ required to reach $s_i^{final}$ at the final step.
  - $s_i^{max}$ : The maximal state of charge.
- Parameters introduced in the function $h_i$:
  - $\delta_t$ : The duration between two successive time steps.
  - $\pi_t^{elec}$ : The price of electric production at the time step $t$.
  - $\gamma$ : A coefficient on the penalty of the final state of charge.
- Parameters introduced in the function $f_i$:
  - $y_t^{up}$ : The maximal reserve that may not be used by electric vehicle at the time step $t$.

**Objective function:** For the cost function in ($\mathcal{P}$), we consider

$$g_i(x_i) = \begin{bmatrix} (c_i^1 - d_i^1) - (\hat{c}_i^1 - \hat{d}_i^1) \\ (c_i^2 - d_i^2) - (\hat{c}_i^2 - \hat{d}_i^2) \\ \vdots \\ (c_i^T - d_i^T) - (\hat{c}_i^T - \hat{d}_i^T) \end{bmatrix}$$

$$f(y) = \sum_{t=1}^{T} \left( y_t - \frac{y_t^{up}}{N} \right)^2$$

$$h_i(x_i) = \Big( \sum_{t=1}^{T} (c_i^t - d_i^t)\delta_t \pi_t^{elec} \Big) - s_i^T \gamma \pi_T^{elec}$$

In summary, the objective function writes:

$$\inf_{x \in \mathcal{X}} \sum_{t=1}^{T} \left[ \frac{1}{N} \Big( \sum_{i=1}^{N} \big( c_i^t - d_i^t - \hat{c}_i^t + \hat{d}_i^t \big) - y_t^{up} \Big) \right]^2 + \frac{1}{N} \Big[ \sum_{i=1}^{N} \Big( \sum_{t=1}^{T} (c_i^t - d_i^t)\delta_t \pi_t^{elec} \Big) - s_i^T \gamma \pi_T^{elec} \Big].$$

**Constraints:**

| Baseline | Service | Description |
|---|---|---|
| $s_i^0 = s_{i,init}$ (1) | $\hat{s}_i^0 = s_{i,init}$ (2) | Initial SoC |
| $0 \leq s_i^t \leq s_i^{max}$ (3) | $0 \leq \hat{s}_i^t \leq s_i^{max}$ (4) | Boundary SoC |
| $u_i^t + v_i^t \leq 1$ (5) | $\hat{u}_i^t + \hat{v}_i^t \leq 1$ (6) | Charge or discharge |
| $c_i^t \leq c_i^{max} u_i^t$ (7) | $\hat{c}_i^t \leq c_i^{max} \hat{u}_i^t$ (8) | Upper bound(c) |
| $d_i^t \leq d_i^{max} v_i^t$ (9) | $\hat{d}_i^t \leq d_i^{max} \hat{v}_i^t$ (10) | Upper bound(d) |
| $u_i^t c_i^{min} \leq c_i^t$ (11) | $\hat{u}_i^t c_i^{min} \leq \hat{c}_i^t$ (12) | Lower bound(c) |
| $v_i^t d_i^{min} \leq d_i^t$ (13) | $\hat{v}_i^t d_i^{min} \leq \hat{d}_i^t$ (14) | Lower bound(d) |
| $s_i^{t+1} = s_i^t + (c_i^{t+1} - d_i^{t+1})$ (15) | $\hat{s}_i^{t+1} = s_i^t + (\hat{c}_i^{t+1} - \hat{d}_i^{t+1})\delta_t$ (16) | Production balance |
| $s_i^t \geq s_i^{t,min}$ (17) | $\hat{s}_i^t \geq s_i^{t,min}$ (18) | Minimum SoC |

Table 1.2: Constraints introduced in the origin problem.

Constraints (1) and (2) dictate the initial state of charge for each electric vehicle. Constraints (3) and (4) bound the state of charge within the permissible range, limited by the vehicle's maximum state of charge and a minimum of 0. Constraints (5) and (6) impose that charging and discharging of an electric vehicle cannot occur simultaneously. Constraints (7)-(14) model the fact that the charging of the vehicle is smaller than $c_i^{max}$ if $u_i^t = 1$, smaller than 0 otherwise. Constraints (15) and (16) specify that the change in charge level between two consecutive time steps should be equal to the difference between production and consumption. Constraints (17) and (18) specify that the state of charge of an vehicle should be greater than the minimum state of charge with which it can reach the expected state of charge at $T$.

# Chapter 2

# State-of-the-art

The algorithmic developments proposed in this report mainly rely on the Stochastic Frank-Wolfe (SFW) algorithm proposed in the article [1]. The specificity of the Algorithm SFW proposed in [1] is that it can deal with the non-convexity of the cost function $J$ (only the function $f$ needs to be convex). It is an iterative algorithm which only requires to solve sub-problems, specific to each agent, of moderate complexity.

## 2.1 Frank-Wolfe Algorithm

The Frank&Wolfe algorithm [2] is an iterative optimization algorithm to resolve convex problem which was first introduced by Marguerite Frank and Philip Wolfe in 1956 in the article [3]. For the Frank&Wolfe algorithm, we consider the following problem:

$$\inf_{x\in\mathbb{R}^N} F(x), \text{ s.t: } x \in S \qquad (\mathcal{P}_{\mathrm{FW}})$$

To apply the Frank&Wolfe algorithm, we require that $f$ is convex and differentiable, that $\nabla f$ is Lipschitz-continuous, and that $S$ is compact and convex.

Given $\tilde{x}$, we define the linearized problem by :

$$\inf_{x\in\mathbb{R}^N} \langle \nabla F(\tilde{x}), x\rangle, \text{ s.t: } x \in S \qquad (\mathcal{P}_{\mathrm{FW},lin}(\tilde{x}))$$

The algorithm begins with an initial candidate, followed by iterative steps. During each iteration, we solve the linearized problem and define the next candidate as a convex combination between the previous candidate and the solution to the linearized problem.

**Algorithm 1:** Frank-Wolfe algorithm

**Input:** $\bar{x}^{(0)} \in S$

1 **for** $k \leftarrow 0$ ***to*** $K$ **do**

2 Find a solution $x_k$ to $\mathcal{P}_{FW,lin}(\bar{x}_k)$

3 Set $\delta_k = \frac{2}{k+2}$

4 s Let $\bar{x}^{k+1} = (1-\delta_k)\bar{x}_k + \delta_k x_k$

**Output:** $\bar{x}^K$, the approximate solution

## 2.2 Algorithm SFW

In the context of the vehicles charging problem, we can not apply the algorithm Frank&Wolfe to the problem ($\mathcal{P}$) because it is not differentiable and not convex. The article [1] proposed a relaxation of ($\mathcal{P}$), consisting in substituting the variables $x_i$ by probability distributions on the set $\mathcal{X}_i$. The relaxed problem being convex, it can be addressed with the Frank&Wolfe algorithm. The main idea of the Stochastic Frank-Wolfe (SFW) algorithm for addressing aggregate optimization problems is structured around two key steps:
The first step is the resolution of the linearized problem, which amounts (as explained in [1]) to solve the $N$ following independent sub-problems:

$$\inf_{x_i \in \mathcal{X}_i} \langle \lambda, g_i(x_i) \rangle + h_i(x_i),$$

where

$$\lambda = \nabla f \left( \frac{1}{N} \sum_{i=1}^{N} g_i(\bar{x}_i) \right)$$

and where $\bar{x}$ denotes the current candidate. The second step is to generate multiples solutions: in iterate $k$, we want to generate $n_k$ different drawings of solutions. We have a probability denoted as $\delta_k = 2/(k+2)$, and a random variable $Z_{i,j,k}$ corresponding to the $i$-th vehicle, $j$-th sample, and $k$-th iteration. The state of each agent is determined as follows:

$$\tilde{x}_i^j = \begin{cases} \bar{x}_i^k & \text{if } 0 < Z_{i,j,k} \leq 1 - \delta_k \\ x_i^k & \text{if } 1 - \delta_k < Z_{i,j,k} \leq 1 \end{cases}$$

The SFW algorithm is described in detail in Section A.

### 2.2.1 Sub-problem formulation

We explicit here the computation of $\lambda$ and the subproblems, in the context of the vehicles charging problems of interest. Given a candidate $\bar{x}$, we denote $\bar{y} = \sum_{i=1}^{N} g_i(\bar{x}_i)$. Recall that the functions $g_i(x_i)$ and $h_i(x_i)$ were introduced at page 3. Then $\lambda$ is given by

$$\lambda_t = (2\bar{y}_t - 2y_t^{up}), \quad \text{for } t = 1, \ldots, T.$$

Consequently, the objective function of the subproblems reads:

$$\inf_{x_i \in \mathcal{X}_i} \; 2 \sum_{t=1}^{T} \frac{\bar{y}_t - y_t^{up}}{N} \left( (c_i^t - d_i^t) - (\hat{c}_i^t - \hat{d}_i^t) \right) + \left( \sum_{t=1}^{T} (c_i^t - d_i^t) \delta_t \pi_t^{elec} \right) - s_i^T \gamma \pi_T^{elec}.$$

# Chapter 3

# Algorithmic developments

While the Stochastic Frank&Wolfe algorithm can yield approximate solutions for the vehicle charging problem, an appreciable discrepancy has been observed between the optimal values produced by the solver (CPLEX) and those generated by the Stochastic Frank&Wolfe algorithm. As a result, we intend to employ the Stochastic Frank&Wolfe algorithm for a specified number of iterations and subsequently transition to the classical Frank&Wolfe algorithm (introduced in the section 3.1). The latter involves solving a partial problem by holding binary variables fixed. Additionally, an alternative greedy algorithm was discovered during the internship(introduced in the section 3.2).

## 3.1 Classical Frank&Wolfe algorithm

The central limitation of the Stochastic Frank&Wolfe algorithm lies in its handling of continuous variables associated with specific vehicles within the problem domain. At each time-step, the algorithm provides a choice limited to either the solution from the previous iteration or the solution by resolving the corresponding sub-problem. In contrast, the Classical Frank&Wolfe algorithm, as detailed in Section 2.1, revolves around obtaining a new solution through a convex combination of the previous solution and the solution derived from solving sub-problems. However, to apply the Classical Frank&Wolfe algorithm, it is necessary to fix all the binary variables. By fixing all the binary variables, we transform a subset of the problem into a convex form, thereby enabling the application of the convex combination.

As a result, we have conceived a novel approach: initially executing $n_{\text{pre}}$ iterations of the Stochastic Frank&Wolfe algorithm to ascertain our binary variables. Armed with these fixed binary variables, we subsequently apply the Classical Frank&Wolfe algorithm in pursuit of a localized optimal solution.

**Algorithm 2:** Classical Frank&Wolfe

**Input:** $\bar{x}^{(K_0)}$: vector containing $(\bar{x}_1^{(k)}, \ldots, \bar{x}_i^{(k)}, \ldots, \bar{x}_N^{(k)})$, each representing the state of an agent, solution after $K_0$ iterations of SFW.

**for** $k \leftarrow K_0$ ***to*** $K$ **do**
- Decompose $\bar{x}^k$ into $(\bar{x}^k_{int}, \bar{x}^k_{con})$, corresponding to integer variables and continuous variables.
- Set $\lambda = \nabla f(\frac{1}{N}\sum_{i=1}^{N} g_i(\bar{x}_{i,int}, \bar{x}_{i,con}))$;
- **for** $i \leftarrow 1$ ***to*** $N$ **do**
  - We want to fix $\bar{x}_{i,int}$ and find a solution $x_{i,con}$ to
  $$\inf_{x_{i,con} \in \mathcal{X}_{i,con}} \langle \lambda, g_i(x_{i,int}, x_{i,con}) \rangle + h_i(x_{i,int}, x_{i,con})$$
  - Set $x_i = (\bar{x}_{i,int}, x_{i,con})$
- $x^k = (x_1, \ldots, x_i, \ldots, x_N)$;
- Find $(\delta_{i,k})$, $i = 1, \cdots, N$ ;
- **for** $j \leftarrow 1$ ***to*** $n_k$ **do**
  - **for** $i \leftarrow 1$ ***to*** $N$ **do**
    - $\bar{x}_i^{k+1} = (1 - \delta_k)\bar{x}_i^k + \delta_k x_i^k$;
- $\bar{x}^{k+1} = (\bar{x}_1^{k+1}, \ldots, \bar{x}_i^{k+1}, \ldots, \bar{x}_N^{k+1})$;

**Output:** $\bar{x}^{K+1}$, the approximate solution

### 3.1.1 Choice of parameter $\delta_{i,k}$

$\delta_i^k$: the parameter $\delta_i^k$ is the coefficient for the convex combination.

**CFW1:Classical Frank&Wolfe 1** At first, the expression of $\delta_{i,k}$ is $\delta_{i,k} = 2/(100 + k + 2)$ which is an expression inspired by the function of the probability of replacing the variables of a certain vehicle by the new solution obtain by solving the corresponding sub-problems.

**CFW2:Classical Frank&Wolfe 2** And then, we want to optimize the parameter to accelerate the convergence, which is very important for our algorithm. The optimisation problem of $\delta_{i,k}$ can be expressed as follow:
Let $\bar{x}^k$ denote the continuous variables obtained at iterate $k$ and $x^k$ denote the solution to the linearized problem(only continuous variables are involved). In the classical Frank&Wolfe algorithm, the solution of the next iterate is given by:

$$\bar{x}^{k+1} = (1 - \delta_k)\bar{x}^k + \delta_k x^k,$$

for the continuous variables(the binary variables remain unchanged).
Therefore $\forall i$, $\delta_{i,k} = \delta_k$, with $\delta_k$ solution of

$$\inf_{\delta \in [0,1]} J((1 - \delta)\bar{x}^k + \delta x^k)$$

**CFW3:Classical Frank&Wolfe 3** After the implementation of the algorithm CF2, we have another inspiration: since that in the classical Frank&Wolfe algorithm, we will do the convex combination for each vehicle to obtain our finale solution, and we can therefore

calculate the optimal $\delta$ for each vehicle.
$\forall i$, $\delta_{i,k}$ solution of

$$\inf_{\delta \in [0,1]^N} J((1-\delta_1)\bar{x}_1^k + \delta x_1^k, \cdots, (1-\delta_N)\bar{x}_N^k + \delta x_N^k)$$

## 3.2 Greedy algorithm

Given the inefficiency of the Stochastic Frank&Wolfe algorithm, our objective is to devise an alternative method that not only demonstrates efficiency but also circumvents the reliance on the classical Frank&Wolfe approach. Thus, we propose the development of a heuristic method, a modified version of the original algorithm, which we refer to as the 'Greedy Algorithm' for this particular problem.

The central concept behind the Greedy Algorithm stems from the Stochastic Frank&Wolfe procedure, where we initially compute the linearized problem to derive the solution vector, denoted as $x_k$, encompassing solutions for all vehicles. Subsequently, we generate a specified number of independent samples. During iteration $k$, a sample is created through a loop corresponding to the number of vehicles, and a random number within the interval $[0, 1]$ is generated. If the generated number is less than the probability $\delta_k = \frac{2}{k+2}$, we select the solution $x_i^k$. Otherwise, the sample resorts to the solution $x$.

In this manner, we replace conventional sampling with our proposed Greedy Method:

**Algorithm 3:** Greedy method

**Input:** $\bar{x}^{(k)}$: vector containing $(\bar{x}_1^{(k)}, \ldots, \bar{x}_i^{(k)}, \ldots, \bar{x}_N^{(k)})$, each representing the state of an agent, solution of the previous iterate.
$x^k$: vector containing $(x_1^{(k)}, \ldots, x_i^{(k)}, \ldots, x_N^{(k)})$, each representing the state of an agent, solution of the linearlized problem.
S : the score correspond to the solution$\bar{x}_i^k$.
$\delta^k = \frac{2}{2+k}$.

```
for i ← 1 to N do
    Set x̄^k_new = x̄^k;
    Generate random values Z_{i,k} from a uniform distribution over the interval [0, 1].
    if Z_{i,k} < 1 − δ_k then
        Set x̄^k_{new,i} = x̄^k;
        Calculate S_new the score of the solution x̄^k_{new,i}.
        if S_new < S then
            Set x̄^k = x̄^k_new;
            Set S = S_new;
x̄^{k+1} = x̄^k.
```

**Output:** $\bar{x}^{K+1}$, the approximate solution

# Chapter 4

# Formulation of different problems

After exploring classical algorithms, we have identified an alternative avenue for potential improvement. This involves optimizing the continuous variables associated with the "Service" using a mechanism referred to as a "booster", applied after each solution update. Furthermore, we have incorporated this "booster" into the original problem, effectively transforming it into a new problem referred to as the "reduced problem".

## 4.1 Original problem + booster

Given our objective function composed of two components, denoted as $f(x)$ and $h(x)$, where $f(x)$ can be expressed as $f(y) = \sum_{t=1}^{T}(c_i^t - d_i^t - \hat{c}_i^t + \hat{d}_i^t - \frac{y_t^{up}}{N})^2$, we address the Vehicle Charging Problem by considering two categories of variables: baseline variables and service variables.

The baseline variables $(c_i^t, d_i^t)$ characterize the authentic behavior of the vehicle charging system. In our scenario, this system not only charges electric vehicles but also supplies electricity for other purposes, referred to as "Service," with a maximum demand of $y_t^{up}$. The service variables $(\hat{c}_i^t, \hat{d}_i^t)$ are introduced to simulate situations where we must provide electricity for these other purposes.

When the baseline variables $c_i^t$ and $d_i^t$ are fixed, we can manually adjust the differences between $\hat{c}_i^t$ and $\hat{d}_i^t$ to minimize the discrepancy between $c_i^t - d_i^t - \hat{c}_i^t + \hat{d}_i^t$ and $y_t^{up}$. We call this mechanism "booster" and we want to implement the booster for all the solutions generated in algorithms. Therefore, we call the problem in which we use the booster for the original problem the booster problem.
And we can therefore express the interval of $\hat{c}_i^t - \hat{d}_t^t$ as :

$$\{\hat{c}_i^t - \hat{d}_t^t | \text{such that constraints 8, 10, 12 , 14 and 16 are satisfied}\}$$
$$= [\hat{u}_i^t c_i^{min} - \hat{v}_i^t d_i^{max},\ \hat{u}_i^t c_i^{max} - \hat{v}_i^t d_i^{min}]$$
$$= \{\underline{\theta}_i^t,\ \overline{\theta}_i^t\}$$

And also, we want the term $\hat{c}_i^t - \hat{d}_i^t$ verifies the capacity of charge level constraint:

$$\frac{s_i^{min} - s_i^t}{\delta_t} \leq \hat{c}_i^{t+1} - \hat{d}_i^{t+1} \leq \frac{s_i^{max} - s_i^t}{\delta_t}$$

Through the amalgamation or integration of these two inequalities, we have($\underline{\theta}_i^t$ and $\overline{\theta}_i^t$ denote the lower bound and upper bound respectively):

$$\underline{\theta}_i^t = \max(\frac{s_i^{min} - s_i^t}{\delta_t}, \hat{u}_i^t c_i^{min} - \hat{v}_i^t) \leq \hat{c}_i^{t+1} - \hat{d}_i^{t+1} \leq \min(\frac{s_i^{max} - s_i^t}{\delta_t}, \hat{u}_i^t c_i^{max} - \hat{v}_i^t d_i^{min}) = \overline{\theta}_i^t$$

We also aim to define the lower and upper bounds of the aggregate variable $y_t$, which are derived from the subtraction of the summation of $(c_i^t - d_i^t)$ across all $N$ vehicles from both the summation of $\bar{\theta}_i^t$ and the summation of $\underline{\theta}_i^t$ across all vehicles, respectively :

$$y_t \in [\underline{y_t}, \bar{y}_t]$$

$$\underline{y_t} = (\sum_{i=1}^{N}(c_i^t - d_i^t)) - \sum_{i=1}^{N}\bar{\theta}_i^t,\ \bar{y}_t = (\sum_{i=1}^{N}(c_i^t - d_i^t)) - \sum_{i=1}^{N}\underline{\theta}_i^t$$

There are three distinct cases for

- if $y_t^{up} \in [\underline{y_t}, \bar{y}_t]$

  In order to minimize the disparity between the aggregate and $y_t^{up}$, we set $y_t = y_t^{up}$. Consequently, we can represent the values of $\hat{c}_i^t - \hat{d}_i^t$ as a convex combination of $\underline{\theta}_i^t$ and $\overline{\theta}_i^t$.

  Hence, the coefficient of the convex combination, denoted as $\beta$, can be computed as follows:

  $$\beta = \frac{y_t - \underline{y_t}}{\overline{y_t} - \underline{y_t}}$$

  And we have the expression of the convex combination:

  $$y_t = (1-\beta)\underline{y_t} + \beta\overline{y_t}$$

  $$= \sum_{i=1}^{N}(c_i^t - d_i^t) - \sum_{i=1}^{N}(\overline{\theta}_i^t(1-\beta) + \underline{\theta}_i^t\beta$$

  Therefore

  $$\text{diff} = (1-\beta)\overline{\theta}_i^t + \beta\underline{\theta}_i^t$$

- else if $y_t^{up} < \underline{y_t}$
  In order to minimize the disparity, we set $y_t = \underline{y_t}$, therefore we have

  $$\text{diff} = \hat{c}_i^t - \hat{d}_i^t = \overline{\theta}_i^t$$

- else ($y_t^{up} > \overline{y_t}$)

  In order to minimize the disparity, we set$y_t = \bar{y}_t$, therefore we have

  $$\text{diff} = \hat{c}_i^t - \hat{d}_i^t = \underline{\theta}_i^t$$

Since $\hat{u}_i^t$ and $\hat{v}_i^t$ can not be simultaneously 1, so we pose :

If $\hat{u}_i^t = 0$, then

$$\hat{c}_i^t = 0,\ \hat{d}_i^t = \text{diff}$$

Else

$$\hat{d}_i^t = 0,\ \hat{c}_i^t = \text{diff}$$

Therefore, we successfully obtain the value of $\hat{c}_i^t$ and $\hat{d}_i^t$ which optimize our local solution(By fixing binary variables and $c_i^t/d_i^t$).

## 4.2 Reduced problem

We want to propose a new version of our problem by optimizing "service" variables $\hat{s}_i^t, \hat{c}_i^t$ and $\hat{d}_i^t$. There are two advantages of this formulation of problem:
First, by reducing "service" variables $\hat{s}_i^t, \hat{c}_i^t$ and $\hat{d}_i^t$, the resolution of our sub-problems will be faster.
Second, the reason why we want to reduce these variables is that the former version of the term $f$ is : $f(y) = \sum_{t=1}^{T}(c_i^t - d_i^t - \hat{c}_i^t + \hat{d}_i^t) - \frac{y_t^{up}}{N})^2$ Just like the section improvement explained, the terms $-\hat{c}_i^t + \hat{d}_i^t$ do not have real influence for real vehicle charging situation and since our function $f$ will calculate the squared distance between $c_i^t - d_i^t - \hat{c}_i^t + \hat{d}_i^t)$ and $\frac{y_t^{up}}{N}$, we can therefore replace $-\hat{c}_i^t + \hat{d}_i^t$ by a interval $[\underline{\theta}_i^t, \overline{\theta}_i^t]$ (the definitions of $\underline{\theta}_i^t$ and $\overline{\theta}_i^t$ are introduced before in the section improvement) which are lower and higher bound that $-\hat{c}_i^t + \hat{d}_i^t$ can reach with the respect to the charge/discharge power and the state of charge maximal/minimal of the vehicle.

$$\begin{bmatrix} s_i^t, c_i^t, d_i^t, u_i^t, v_i^t \\ \hat{s}_i^t, \hat{c}_i^t, \hat{d}_i^t, \hat{u}_i^t, \hat{v}_i^t \end{bmatrix} \rightarrow x_i \xrightarrow{X=\prod_{i=0}^{N} x_i} J(X) \qquad \text{(Actual problem)}$$

$$\downarrow$$

$$\begin{bmatrix} s_i^t, c_i^t, d_i^t, u_i^t, v_i^t \\ \hat{u}_i^t, \hat{v}_i^t \end{bmatrix} \rightarrow x_i \xrightarrow{X=\prod_{i=0}^{N} x_i} \tilde{J}(X) \qquad \text{(Reduced problem)}$$

To formulate our new problem, we want to induce a new function $R(y, y_1, y_2)$ where $y_1 < y_2$, and the definition of the function:

$$R(y, y_1, y_2) = \begin{cases} 0 & \text{if } y \in [y_1, y_2] \\ (y - y_2)^2 & \text{if } y \geq y_2 \\ (y - y_1)^2 & \text{if } y \leq y_1 \end{cases}$$

And we want to replace our function $f$ in the objective function by:

$$\alpha \sum_{t=1}^{T} \tilde{f}_t \left( \frac{1}{N} \sum_{i=1}^{N} \tilde{g}_{i,t}^{(1)}(x_i), \frac{1}{N} \sum_{i=1}^{N} \tilde{g}_{i,t}^{(2)}(x_i) \right)$$

$$\text{where } \tilde{g}_{i,t}^{(1)}(x_i) = c_i^t - d_i^t - \min(\hat{u}_i^t c_i^{max} - \hat{v}_i^t d_i^{min}, \frac{s_i^{max} - s_i^{t-1}}{\delta_t})$$
$$\tilde{g}_{i,t}^{(2)}(x_i) = c_i^t - d_i^t - \max(\hat{u}_i^t c_i^{min} - \hat{v}_i^t d_i^{max}, \frac{s_i^{min} - s_i^{t-1}}{\delta_t})$$

which represent respectively the the maximum value and the minimum value of the difference between $c_i^t - d_i^t$ and $\hat{c}_i^t - \hat{d}_i^t$.

So we have:

$$\begin{aligned} \tilde{f}_t(y_1, y_2) &= R\left(\frac{y_t^{up}}{N}, y_1, y_2\right) \\ &= R\left(\frac{y_t^{up}}{N}, \frac{1}{N}\sum_{i=1}^{N}(c_i^t - d_i^t - \overline{\theta}_i^t), \frac{1}{N}(\sum_{i=1}^{N}(c_i^t - d_i^t - \underline{\theta}_i^t)\right) \end{aligned}$$

So the objective function can be written as :

$$\inf_{x=(x_1,\ldots,x_N), x_i \in \mathcal{X}_i} \alpha \sum_{t=1}^{T} \tilde{f}_t \left( \frac{1}{N} \sum_{i=1}^{N} \tilde{g}_{i,t}^{(1)}(x_i), \frac{1}{N} \sum_{i=1}^{N} \tilde{g}_{i,t}^{(2)}(x_i) \right) + \frac{1}{N} \sum_{i=1}^{N} h_i(x_i) \qquad (P_R)$$

By eliminating "service" variables $\hat{s}_i^t, \hat{c}_i^t$ and $\hat{d}_i^t$ and using the expression $\underline{\theta}$ and $\overline{\theta}$, we have not only simplified the problem, but also accelerated the process of optimization because the optimization of function $f$ is done by construction of the problem.

### 4.2.1 Sub-problem formulation of the reduced problem

Since we have formulated our reduced problem, to apply our algorithm stochastic Frank&Wolf, we need to formulate our sub-problems associated to each electric vehicle. From the previous formulation, we knows that the sub-problem is expressed in the form of the sum of the $h_i$ function and the scalar product of the derivative of function $f$ at the previous point(solution of the previous iteration) and the term $g_i(x_i)$ which represent the aggregative sum.

Here, we consider $k$ as the iteration number and $\bar{x}_i^k$ represents , the solution of the previous iteration which can help us to calculate the derivative $\lambda_t^{(1)}$ and $\lambda_t^{(2)}$ where the (1) and (2) in the exponent position of $\lambda_t^{(1)}$ represents the variable corresponding to differentiation.

And also, we have $x_i^k$ which represents the new solution that we want to obtain after the optimization step.

$$\text{We note} \qquad \lambda_t^{(1)} = \frac{\partial \hat{f}_t(\bar{y}_t^{(1),k}, \hat{y}_t^{(2),k})}{\partial y_1} = \frac{\partial R(\frac{y_t^{up}}{N}, \bar{y}_t^{(1),k}, \bar{y}_t^{(2),k})}{\partial y_1},$$

$$\lambda_t^{(2)} = \frac{\partial \hat{f}_t(\bar{y}_t^{(1),k}, \bar{y}_t^{(2),k})}{\partial y_2} = \frac{\partial R(\frac{y_t^{up}}{N}, \bar{y}_t^{(1),k}, \bar{y}_t^{(2),k})}{\partial y_2}.$$

$$\text{with} \qquad \bar{y}_t^{(1),k} = \frac{1}{N}\sum_{i=1}^{N} \tilde{g}_{i,t}^{(1)}(\bar{x}_i^k) \text{ and } \bar{y}_t^{(2),k} = \frac{1}{N}\sum_{i=1}^{N} \tilde{g}_{i,t}^{(2)}(\bar{x}_i^k).$$

So our sub-problem can be written as :

$$\inf_{x_i \in \mathcal{X}_i} \alpha \sum_{t=1}^{T} (\frac{1}{N}(\lambda_t^{(1)} \tilde{g}_{i,t}^{(1)}(\bar{x}_i^k) + \lambda_t^{(2)} \tilde{g}_{i,t}^{(2)}(\bar{x}_i^k)) + \sum_{i=1}^{N} h_i(x_i^k)$$

We have found out that the derivative of the function $R$ with respect to the variable $y_1$ is always positive and always negative to the variable $y_2$:

$$\frac{\partial R(y, y_1, y_2)}{\partial y_1} = \begin{cases} 0 & \text{if } y \in [y_1, y_2] \\ 0 & \text{if } y \geq y_2 \\ 2(y_1 - y) & \text{if } y \leq y_1 \end{cases}$$

$$\frac{\partial R(y, y_1, y_2)}{\partial y_2} = \begin{cases} 0 & \text{if } y \in [y_1, y_2] \\ 2(y_2 - y) & \text{if } y \geq y_2 \\ 0 & \text{if } y \leq y_1 \end{cases}$$

because we have $y_1 \leq y \leq y_2$, so $-2(y_1 - y) \geq 0$ and $-2(y_2 - y) \leq 0$.

We want to add variables $z_{i,t}^{(1)}$ and $z_{i,t}^{(2)}$ to represent the terms $\min(\hat{u}_i^t c_i^{max} - \hat{v}_i^t d_i^{min}, \frac{s_i^{max} - s_i^{t-1}}{\delta_t})$ and $\max(\hat{u}_i^t c_i^{min} - \hat{v}_i^t d_i^{max}, \frac{s_i^{min} - s_i^{t-1}}{\delta_t})$:

$$\begin{aligned} z_{i,t}^{(1)} &\geq -\hat{u}_i^t c_i^{max} + \hat{v}_i^t d_i^{min} \\ z_{i,t}^{(1)} &\geq \frac{-s_i^{max} + s_i^{t-1}}{\delta_t} \\ z_{i,t}^{(2)} &\geq \hat{u}_i^t c_i^{min} - \hat{v}_i^t d_i^{max} \\ z_{i,t}^{(2)} &\geq \frac{s_i^{min} - s_i^{t-1}}{\delta_t} \end{aligned}$$

So our sub-problem can be also written as (here $c_i^t$ and $d_i^t$ correspond to $\bar{x}_i^k$ ):

$$\inf_{x_i \in \mathcal{X}_i} \alpha \sum_{t=1}^{T} \left( \frac{1}{N} (\lambda_t^{(1)} c_i^t - \lambda_t^{(1)} d_i^t + \lambda_t^{(1)} z_t^{(1)} + \lambda_t^{(2)} c_i^t - \lambda_t^{(2)} d_i^t - \lambda_t^{(2)} z_t^{(2)}) \right) + \sum_{i=1}^{N} h_i(x_i^k)$$

### 4.2.2 Another formulation of the reduced problem

From the previous sections, we know that the function $R(y, y_1, y_2)$ is increasing with respect to the variable $y_1$ and decreasing with respect to the variable $y_2$ since previously we have $\frac{\partial R}{\partial y_1} \geq 0$ and $\frac{\partial R}{\partial y_2} \leq 0$.

We want to replace the term of $\min(\hat{u}_i^t c_i^{max} - \hat{v}_i^t d_i^{min}, \frac{s_i^{max} - s_i^{t-1}}{\delta_t})$ and $\max(\hat{u}_i^t c_i^{min} - \hat{v}_i^t d_i^{max}, \frac{s_i^{min} - s_i^{t-1}}{\delta_t})$ directly by $z_{i,t}^{(1)}$ and $z_{i,t}^{(2)}$. Our first formulation of the reduced problem :

$$\inf_{x=(x_1,\dots,x_N), x_i \in \mathcal{X}_i} \alpha \sum_{t=1}^{T} \tilde{f}_t \left( \frac{1}{N} \sum_{i=1}^{N} \tilde{g}_{i,t}^{(1)}(x_i), \frac{1}{N} \sum_{i=1}^{N} \tilde{g}_{i,t}^{(2)}(x_i) \right) + \frac{1}{N} \sum_{i=1}^{N} h_i(x_i) \qquad (P_R)$$

$$\downarrow \text{turns into}$$

$$\inf_{x_i \in \mathcal{X}_i} \alpha \sum_{t=1}^{T} \tilde{f}_t \left( \frac{1}{N} \sum_{i=1}^{N} (c_i^t - d_i^t + z_{i,t}^{(1)}), \frac{1}{N} \sum_{i=1}^{N} (c_i^t - d_i^t - z_{i,t}^{(2)}) \right) + \frac{1}{N} \sum_{i=1}^{N} h_i(\bar{x}_i^k) \qquad (P_R')$$

And respectively these constraints characterizing $z_{i,t}^{(1)}$ and $z_{i,t}^{(2)}$ are added:

$$\begin{aligned}
z_{i,t}^{(1)} &\geq -\hat{u}_i^t c_i^{max} + \hat{v}_i^t d_i^{min} \\
z_{i,t}^{(1)} &\geq \frac{-s_{i,t}^{max} + s_i^{t-1}}{\delta_t} \\
z_{i,t}^{(2)} &\geq \hat{u}_i^t c_i^{min} - \hat{v}_i^t d_i^{max} \\
z_{i,t}^{(2)} &\geq \frac{s_i^{min} - s_i^{t-1}}{\delta_t}
\end{aligned}$$

Compare to the problem $P_R$, our newly formulated problem $P_R'$ can not only apply our stochastic Frank&Wolfe algorithm, but also our standard Frank&Wolfe algorithm because of replacing expressions of max and min by variables.

# Chapter 5

# Numerical evaluation

From the model part, we know that we have implemented three different problems:

- P1 : original problem
- P2 : original+booster problem
- P3 : reduced problem

And we have stochastic Frank-Wolfe algorithms SFW1(the basic algorithm) and SFW2(Greedy algorithm).

- SFW1 : stochastic Frank&Wolfe as described in [1] (10 samples, 100 iterations)
- SFW2 : greedy research (100 iterations)

We employ three distinct variants of the classical Frank-Wolfe algorithm, each distinguished by the method used to compute the coefficient $\delta_k$ for the convex combination (10 samples, 50 iterations of SFW and 50 iterations of classical FW):

- CFW1 : $\delta_k = 2/(p+k)$, with $p \geq 2$
- CFW2 : calculate the optimal $\delta_k$
- CFW3 : calculate the optimal $\delta_{i,k}$

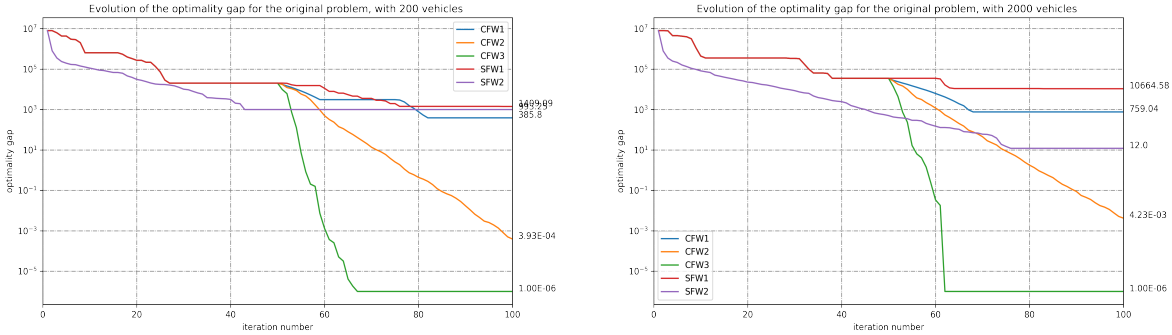
**Optimality Gap**: In our tests, we compare the best optimal value to $1e^{-6}$ to facilitate logarithmic-scale visualization and select the maximum of these two values.

## 5.1 First test : Comparison by iterations

I aim to assess the robustness and efficacy of all methodologies. To achieve this, I have generated distinct instances involving varying numbers of vehicles (50, 100, 200, 500, 1000, and 2000). For each instance, I have adapted it to address three distinct questions, thereby producing different graphs for each question.

For clarity, this presentation focuses only on the examination of graphs associated with instances of sizes 200 and 2000. Additional graphs for other instance sizes are provided in the appendix B. Here are the initial findings:

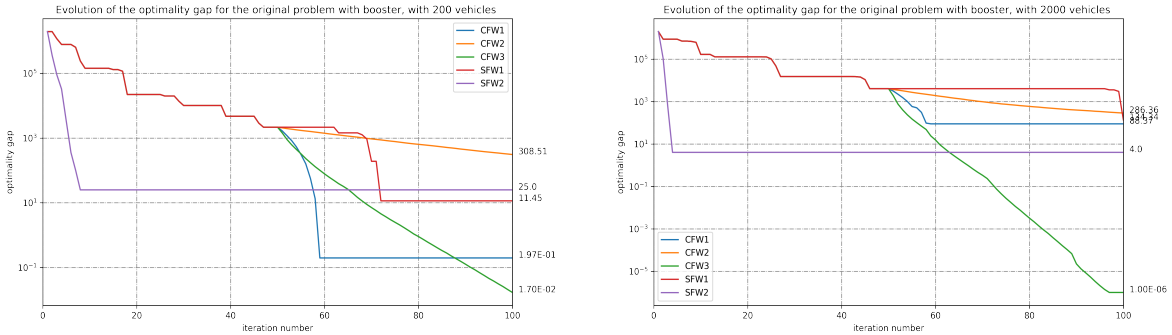
**Original Problem:**




(a) nb=200 (b) nb=2000

Figure 5.1: The graphs of the original problem exhibit variations in performance across different vehicle sizes. Notably, the CFW3 algorithm consistently outperforms other algorithms in solving the original problem, consistently yielding optimal solutions for these instances. Additionally, the order CFW3 > CFW2 > CFW1 aligns with their respective definitions, confirming the intuitive expectations.

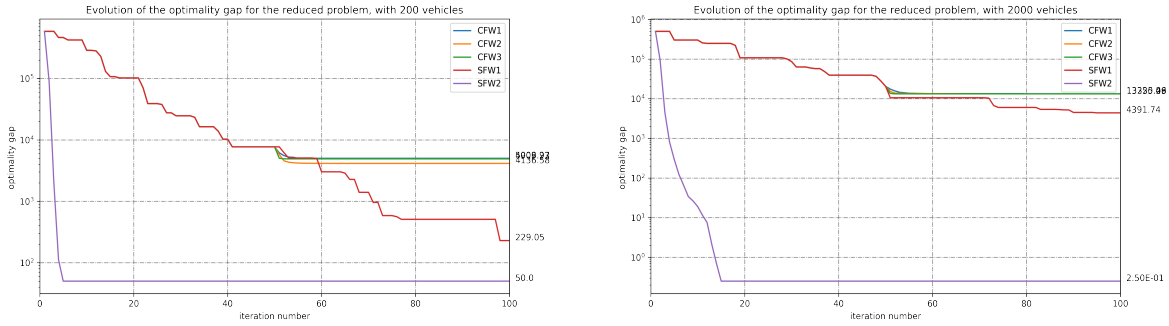
**Original+booster Problem:**




(a) nb=200 (b) nb=2000

Figure 5.2: The graphs of the original+booster problem exhibit variations in performance across different vehicle sizes. In this problem, it is noteworthy that the Classical Frank & Wolfe algorithm does not follow a fixed order due to the incorporation of the booster. Initially, CFW3 may converge to a superior solution, but the application of the booster may enable other algorithms to discover better solutions, potentially within regions that accelerate convergence. And it is observed that, notably, the SFW2 algorithm exhibits rapid convergence, achieving its objective in fewer than 10 iterations.

**Reduced Problem:**




(a) nb=200 (b) nb=2000

Figure 5.3: The graphs of the reduced problem exhibit variations in performance across different vehicle sizes. It is evident that the SFW2 algorithm outperforms all other methods, converging rapidly within fewer than 10 iterations. Furthermore, the observed order of performance is CFW3 > CFW2 > CFW1. This order is attributed to the incorporation of the 'booster' mechanism into the problem, as every solution returned is already optimized due to this inclusion.

**Remark 1** Initially, based on intuition, we posit that Algorithm CFW3 outperforms both CFW1 and CFW2, a trend consistently observed across all three problems. Notably, this observation holds true for both the original problem and its reduced variant. However, it's worth noting that in some instances, CFW2 exhibits superior performance compared to CFW3, driven by specific properties inherent to the original+booster problem.
**Remark 2** It is noteworthy that the SFW2 algorithm consistently exhibits rapid convergence and demonstrates superior performance in addressing the reduced problem..

## 5.2 Robust algorithm : a combination of SFW2 and CFW3

After the initial numerical evaluation, we have made several observations. Building upon Remarks 1 and 2 in the subsequent section, an idea has emerged: combining SFW2 and CFW3 to form a novel algorithm, hereafter referred to as the 'Robust Algorithm.'

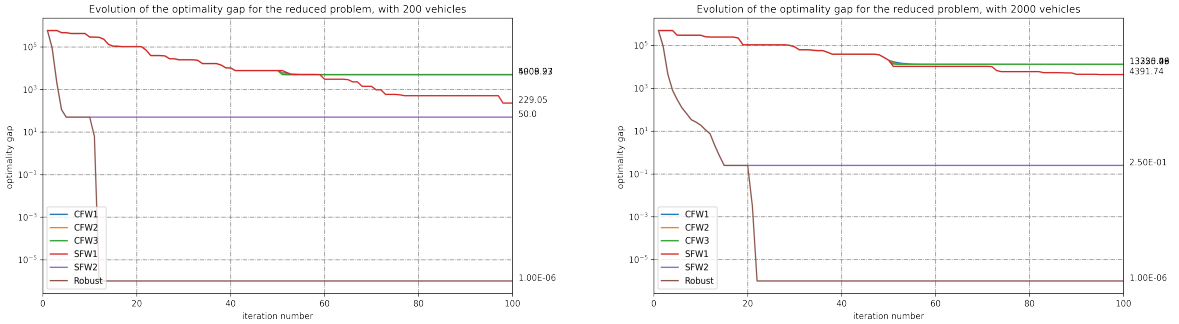
**Primary Concept:** Given the rapid convergence characteristics of the SFW2 algorithm during its initial iterations, we propose commencing the optimization process with SFW2. If no improvement is observed over a span of 5 iterations, we will transition to the CFW3 algorithm, which is anticipated to yield superior solutions.

This algorithm is named the 'Robust Algorithm' due to its derivation from our two leading algorithms, SFW3 and CFW3, and its remarkable performance in terms of minimizing the optimality gap and runtime.

## 5.3 Second test: Comparison by iterations

I want to do the same tests as first test but adding the robust algorithm:

**Reduced Problem with robust algorithm:**

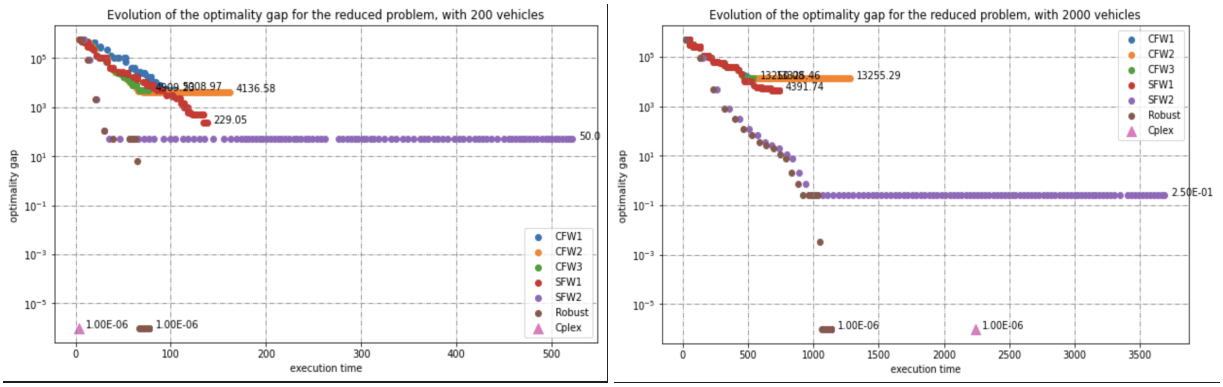



(a) nb=200 (b) nb=2000

Figure 5.4: Having established the superior performance of SFW2 in the previous test on the reduced problem, we proceeded to compare our newly implemented 'Robust Algorithm' with SFW2. Initially, both algorithms exhibited a similar behavior, descending rapidly within the first 10 to 20 iterations before reaching a convergence plateau. However, the Robust Algorithm, designed to enhance robustness, subsequently switched to CFW3 mode after detecting no improvements for five consecutive iterations. And remarkably, our Robust Algorithm achieved the optimal solution after only 20 iterations.

## 5.4 Comparison by time

Our "robust algorithm" consistently converges efficiently, minimizing the required iterations for satisfactory solutions. Our objective is to optimize computational efficiency and compare it to the CPLEX solver's execution time.




(a) nb=200 (b) nb=2000

Figure 5.5: These graphs depict the execution times. It is observed that the point corresponding to the final iteration of the robust algorithm dominates all other 'final' points from different algorithms. This observation suggests that the robust algorithm exhibits superior efficiency. Additionally, it is noteworthy that the final point of the robust algorithm lies below, or at the very least, equals all other data points. This finding implies that the robust algorithm consistently provides the best solution. The CPLEX solver demonstrates remarkable efficiency when handling small-scale instances. However, when confronted with larger instances involving 2000 vehicles, the execution time of CPLEX becomes significantly slower than our robust algorithm.

# Chapter 6

# Conclusion

During the course of the internship, a sequence of algorithmic improvements was undertaken. Initially, the Stochastic Frank&Wolfe algorithm was implemented. Subsequently, a more efficient heuristic, the Greedy Algorithm, was developed by replacing the conventional sampling procedure with a greedy comparison mechanism that leverages the previous solution and the newly generated solution.

Furthermore, a combination of the Stochastic Frank&Wolfe algorithm and a Frank&Wolfe algorithm adapted to a partial problem resulted in the formulation of the Classical Frank&Wolfe algorithm. This classical variant consistently yields solutions with a specified level of precision.

Lastly, by amalgamating the advantages of both SFW2 and CFW3, a robust algorithm was derived. This algorithm consistently produces satisfactory solutions and demonstrates remarkable computational efficiency, particularly when applied to large-scale problems.

We conducted tests on all algorithms using relatively straightforward instances. For future investigations, it is advisable to consider the evaluation of our algorithms on more complex instances characterized by extended time-steps, diverse vehicle profiles, and greater variability in electricity prices. This approach will allow for a more comprehensive examination of the behaviors exhibited by our algorithms.

While our existing 'Robust' algorithm successfully resolves all tested instances within our current context, we acknowledge the potential existence of more challenging instances that may elude its perfect solution. To address this, we propose the implementation of new algorithms designed to tackle such scenarios. Specifically, we envision an algorithmic approach that alternates between "SFW2" and "CFW3" iterations (SFW2 -> CFW3 -> SFW2 -> $\cdots$).

In addition, optimizing the code is a viable option. For instance, we can leverage parallelization mechanisms to concurrently compute all sub-problems. Furthermore, it is feasible to implement the algorithm in another programming language that is better suited for optimization problems, such as Julia.

# Appendix A

# Stochastic Frank-Wolfe algorithm

**Algorithm 4:** Stochastic Frank-Wolfe Algorithm

**Input:** $\bar{x}^{(0)}$: vector containing $(\bar{x}_1^{(0)}, \ldots, \bar{x}_i^{(0)}, \ldots, \bar{x}_N^{(0)})$, each representing the state of an agent

$(n_k)_{k \in N}$ : a list of integers, where $n_k$ represents the number of drawings for the $k$-th iteration.

**for** $k \leftarrow 1$ ***to*** $K$ **do**
  Set $\lambda = \nabla f(\frac{1}{N} \sum_{i=1}^{N} g_i(\bar{x}_i))$;
  **for** $i \leftarrow 1$ ***to*** $N$ **do**
    Find a solution $x_i^k$ to

$$\inf_{x_i^k \in X_i} \langle \lambda, g_i(x_i) \rangle + h_i(x_i)$$

  $x^k = (x_1^k, \ldots, x_i^k, \ldots, x_N^k)$;
  Set $\delta_k = 2/(k+2)$ ;
  **for** $j \leftarrow 1$ ***to*** $n_k$ **do**
    **for** $i \leftarrow 1$ ***to*** $N$ **do**
      Draw the state of each agent $\tilde{x}_i^j$ according to

$$\tilde{x}_i^j = \begin{cases} \bar{x}_i^k & \text{if } 0 < Z_{i,j,k} \leq 1 - \delta_k \\ x_i^k & \text{if } 1 - \delta_k < Z_{i,j,k} \leq 1, \end{cases}$$

      Here $Z_{i,j,k}$ is a random variable with uniform distribution in $[0, 1]$.
    Set $\tilde{x}^j = (\tilde{x}_1^j, \ldots, \tilde{x}_i^j, \ldots, \tilde{x}_N^j)$;
  Find $\bar{x}^{(k+1)} \in \arg\min_{\forall j \in \{1, \cdots, nk\}, x \in \{\tilde{x}_1^j, \ldots, \tilde{x}_N^j\}} J(x)$ ;

**Output:** $\bar{x}^{K+1}$, the approximate solution

**Algorithm 5:** Classical Frank-Wolfe Algorithm

**Input:** $\bar{x}^{(0)}$: vector containing $(\bar{x}_1^{(0)}, \ldots, \bar{x}_i^{(0)}, \ldots, \bar{x}_N^{(0)})$, each representing the state of an agent

$(n_k)_{k\in N}$ : a list of integers, where $n_k$ represents the number of drawings for the $k$-th iteration.

$n_{pre}$ : a number representing the number of iterations of stochastic FW algorithm to be done at the beginning of the algorithm.

**for** $k \leftarrow 1$ ***to*** $n_{pre}$ **do**
- Set $\lambda = \nabla f(\frac{1}{N}\sum_{i=1}^{N} g_i(\bar{x}_i))$;
- **for** $i \leftarrow 1$ ***to*** $N$ **do**
  - Find a solution $x_i^k$ to
$$\inf_{x_i^k \in X_i} \langle \lambda, g_i(x_i)\rangle + h_i(x_i)$$
- $x^k = (x_1^k, \ldots, x_i^k, \ldots, x_N^k)$;
- Set $\delta_k = 2/(k+2)$ ;
- **for** $j \leftarrow 1$ ***to*** $n_k$ **do**
  - **for** $i \leftarrow 1$ ***to*** $N$ **do**
    - Draw the state of each agent $\tilde{x}_i^j$ according to
$$\tilde{x}_i^j = \begin{cases} \bar{x}_i^k & \text{if } 0 < Z_{i,j,k} \leq 1-\delta_k \\ x_i^k & \text{if } 1-\delta_k < Z_{i,j,k} \leq 1, \end{cases}$$
Here $Z_{i,j,k}$ is a random variable with uniform distribution in $[0,1]$.
    - Set $\tilde{x}^j = (\tilde{x}_1^j, \ldots, \tilde{x}_i^j, \ldots, \tilde{x}_N^j)$;
  - Find $\bar{x}^{(k+1)} \in \arg\min_{\forall j\in\{1,\cdots,nk\}, x\in\{\tilde{x}_1^j,\ldots,\tilde{x}_N^j\}} J(x)$ ;

**for** $k \leftarrow n_{pre}$ ***to*** $K$ **do**
- Decompose $\bar{x}^k$ into $(\bar{x}_{int}^k, \bar{x}_{con}^k)$, corresponding to integer variables and continuous variables.
- Set $\lambda = \nabla f(\frac{1}{N}\sum_{i=1}^{N} g_i(\bar{x}_{i,int}, \bar{x}_{i,con}))$;
- **for** $i \leftarrow 1$ ***to*** $N$ **do**
  - We want to fix $\bar{x}_{i,int}$ and find a solution $x_{i,con}$ to
$$\inf_{x_{i,con}\in X_{i,con}} \langle \lambda, g_i(x_{i,int}, x_{i,con})\rangle + h_i(x_{i,int}, x_{i,con})$$
Set $x_i = (\bar{x}_{i,int}, x_{i,con})$
- $x^k = (x_1, \ldots, x_i, \ldots, x_N)$;
- Set $\delta_k = 2/(k - n_{pre} + 2)$ ;
- **for** $j \leftarrow 1$ ***to*** $n_k$ **do**
  - **for** $i \leftarrow 1$ ***to*** $N$ **do**
    - $\bar{x}_i^{k+1} = (1-\delta_k)\bar{x}_i^k + \delta_k x_i^k$;
- $\bar{x}^{k+1} = (\bar{x}_1^{k+1}, \ldots, \bar{x}_i^{k+1}, \ldots, \bar{x}_N^{k+1})$;
- **Output:** $\bar{x}^{K+1}$, the approximate solution

**Algorithm 6:** Resolution of the linear programming problem

**Input:** $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_i, \ldots, \bar{x}_N)$

Set $\lambda = \nabla f(\frac{1}{N}\sum_{i=1}^{N} g_i(\bar{x}_{i,int}, \bar{x}_{i,con}))$;

**for** $i \leftarrow 1$ ***to*** $N$ **do**

We want to fix $\bar{x}_{i,int}$ and find a solution $x_{i,con}$ to

$$\inf_{x_{i,con} \in X_{i,con}} \langle \lambda, g_i(x_{i,int}, x_{i,con}) \rangle + h_i(x_{i,int}, x_{i,con})$$

Set $x_i = (\bar{x}_{i,int}, x_{i,con})$

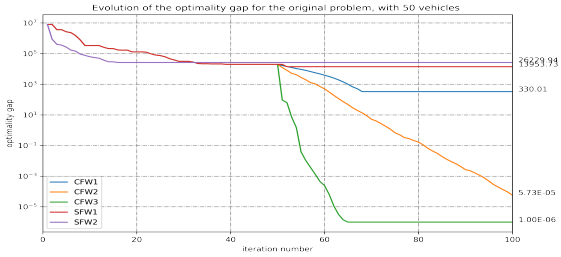
**Output:** $(x_1, \ldots, x_i, \ldots, x_N)$

# Appendix B

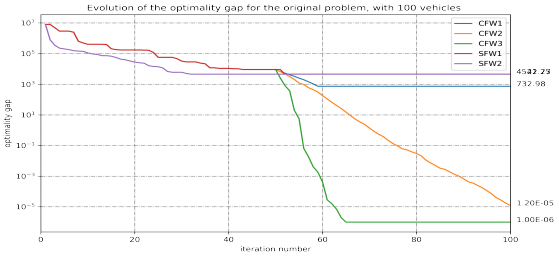
# Graphs

This chapter presents a comprehensive collection of graphs obtained from instances ranging in size from 50 to 2000, where the size refers to the number of vehicles.

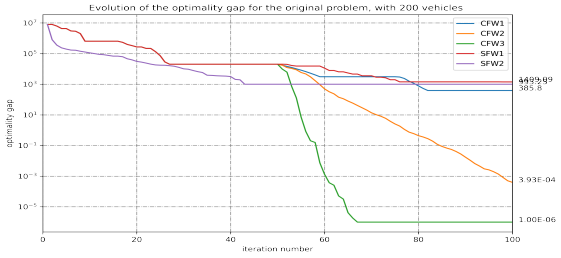
## B.1 Graphs by iterations



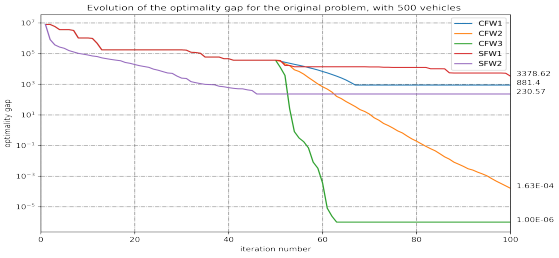

(a) nb=50



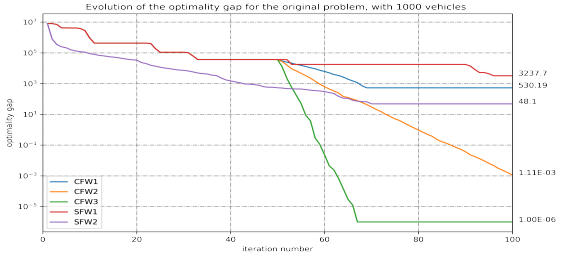

(b) nb=100



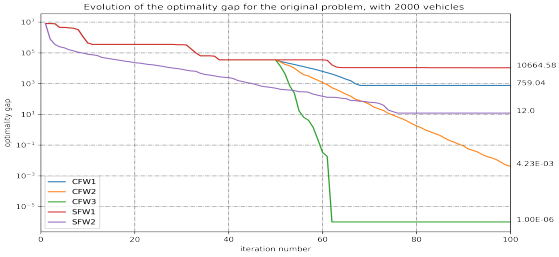

(c) nb=200




(d) nb=500

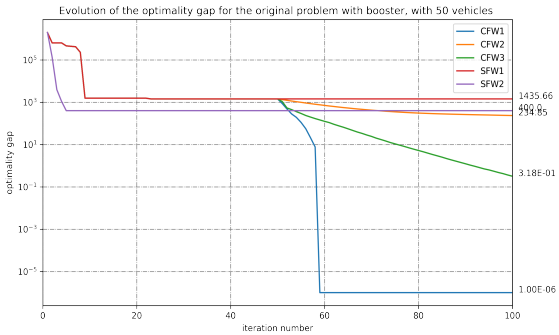



(e) nb=1000

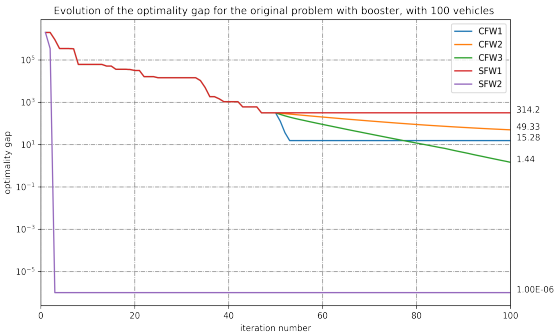



(f) nb=2000

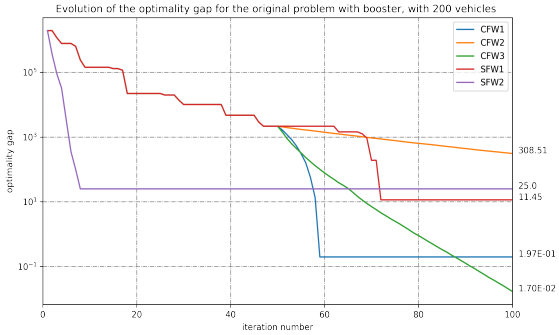
Figure B.1: The graphs of original problem with differences sizes of vehicles.



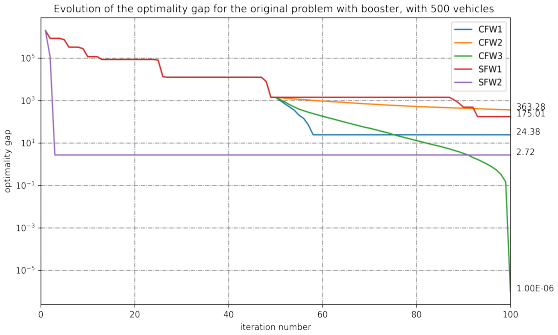

(a) nb=50



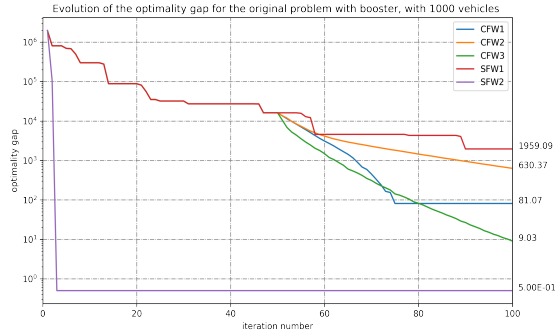

(b) nb=100



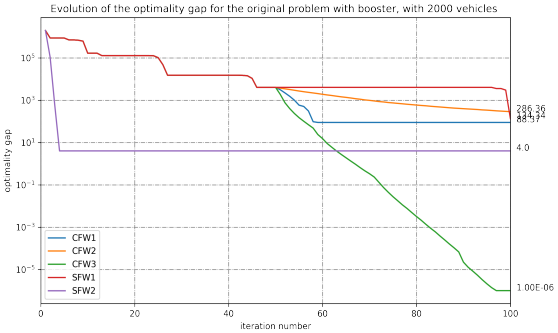

(c) nb=200




(d) nb=500

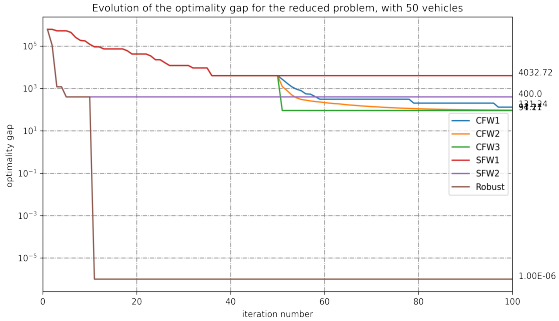



(e) nb=1000

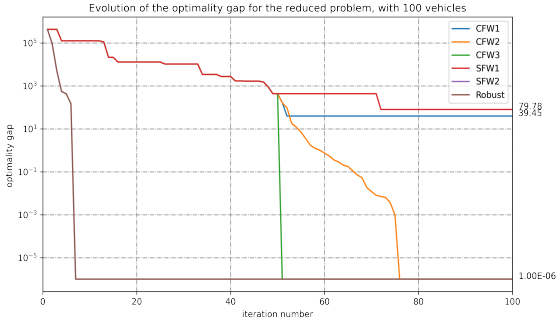



(f) nb=2000

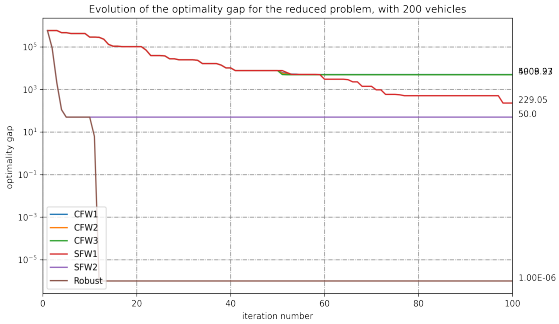
Figure B.2: The graphs of original problem + booster with differences sizes of vehicles.



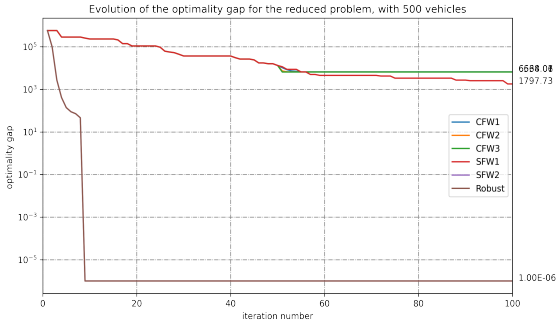

(a) nb=50



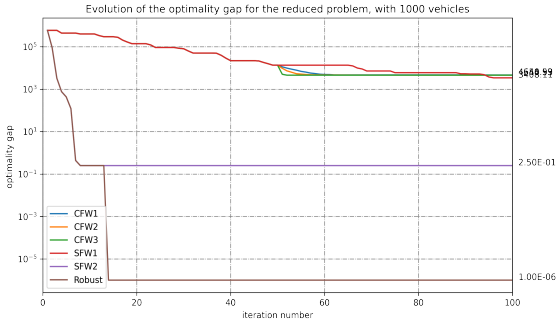

(b) nb=100



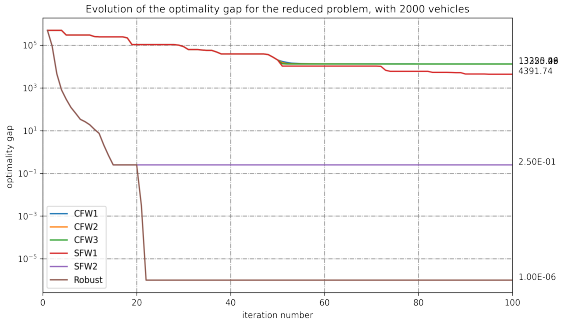

(c) nb=200




(d) nb=500



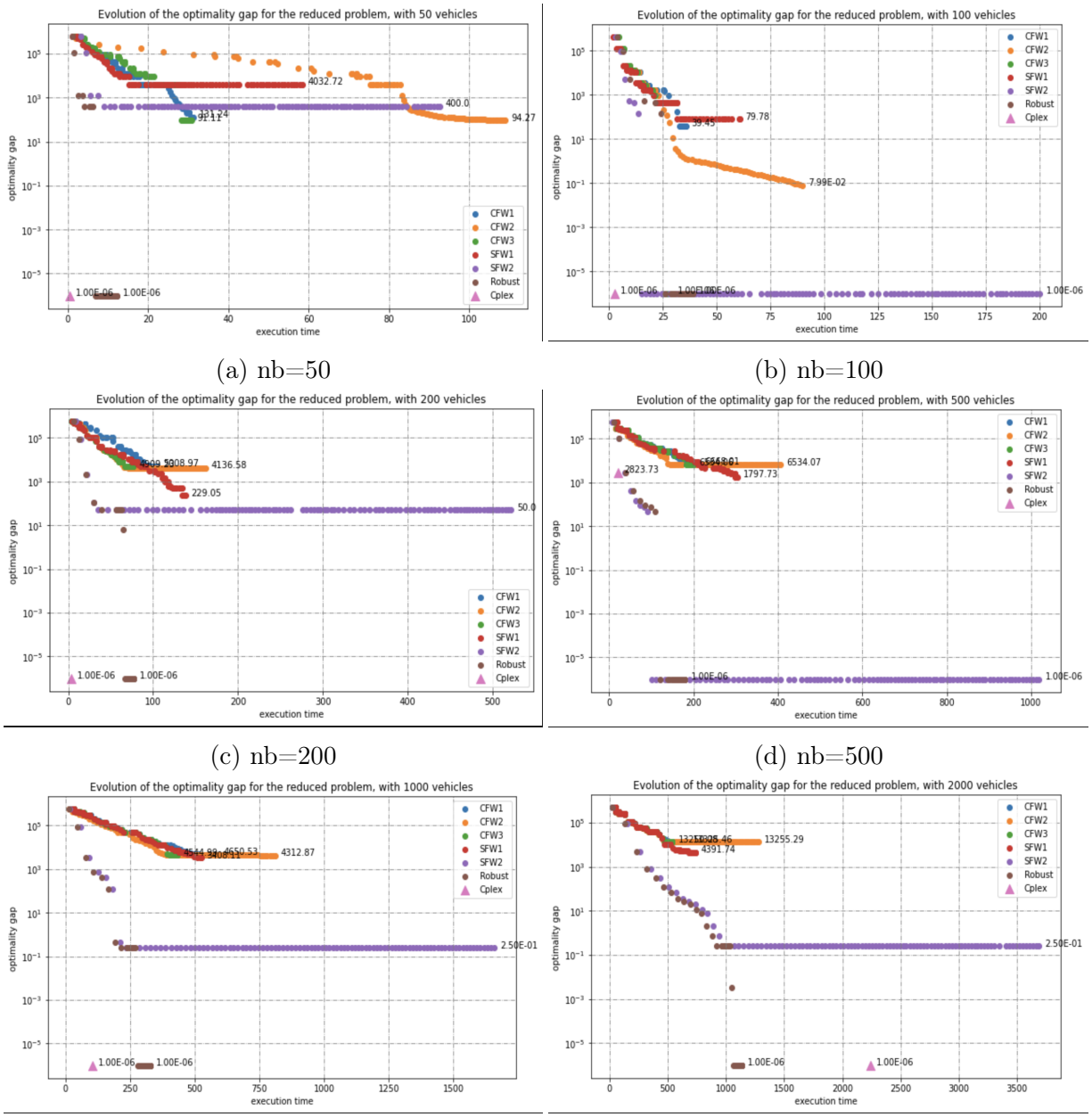

(e) nb=1000




(f) nb=2000

Figure B.3: The graphs of reduced problem with differences sizes of vehicles.

## B.2 Execution time by iterations (Reduced problem)




(a) nb=50

(b) nb=100

(c) nb=200

(d) nb=500

(e) nb=1000

(f) nb=2000

Figure B.4: The graphs of reduced problem with differences sizes of vehicles.

# Appendix C

# MIQP

In the part.6 of Kang Liu's article, he provided numerical results for a mixed-integer linear quadratic problem of the form (P). With $A$ a real $M \times N$ matrix and $\bar{y} \in \mathbb{R}^M$, the problem can be written as follows:

$$(\text{MIQP}) \min_{x \in \{0,1\}^N} J(x) := \frac{1}{N^2} \|Ax - \bar{y}\|^2_{\mathbb{R}^M} = \sum_{j=1}^{M} \left( \frac{1}{N} \sum_{i=1}^{N} A_{ji} x_i - \frac{\bar{y}_j}{N} \right)^2 .$$

Following the same parameters setting :

- M=N,
- $\forall i \in N, j \in M, A_{ji} \in [0, 1]$,
- $\forall i \in N, \bar{y}_i \in [0, N/2]$,
- $K = 2N$,
- $\forall k \in K, n_k = 1$.

Kang's first experiment is a comparison of Algorithm Frank-Wolfe with two solvers, SCIP and GUROBI. Here i only compare the algorithm with GUROBI to verify the validity of our algorithm in terms of times and accuracy of solutions.

Here, the way to calculate the gap is a little different from Kang's formulation, i use $gap = (V_s - V_g)/V_g$ rather than $gap = (V_s - \mathcal{J}^*)/\mathcal{J}^*$(Kang's definition) .
($\mathcal{J}^*$ the optimal value,$V_s$ the value returned by the algorithm, $V_g$ the value returned by the resolution of GUROBI).

| N=M | GUR | SFW | | GUR | SFW |
|---|---|---|---|---|---|
| | value | value | gap | time in seconds | |
| 100 | 1.905 | 1.913 | 0.409% | 0.85 | 0.03 |
| 200 | 3.142 | 3.147 | 0.181% | 5.33 | 0.06 |
| 400 | 7.461 | 7.469 | 0.117% | 42.99 | 0.14 |
| 800 | 16.155 | 16.161 | 0.042% | 1286.32 | 1.18 |
| 1600 | Nan | 30.703 | Nan | Nan | 9.27 |
| 3200 | Nan | 65.337 | Nan | Nan | 89.37 |

Table C.1: My results.(On a laptop with Intel Core i5(dual-core), 2.3GHz, 16GB RAM)

| N=M | GUR | SFW | | GUR | SFW |
|---|---|---|---|---|---|
| | value | value | gap | time in seconds | |
| 100 | 2.077 | 2.136 | 2.870% | 0.20 | 0.03 |
| 200 | 4.120 | 4.159 | 0.956% | 0.69 | 0.09 |
| 400 | 7.871 | 7.904 | 0.430% | 7.90 | 0.91 |
| 800 | 15.954 | 15.966 | 0.079% | 10.63 | 6.18 |
| 1600 | 32.048 | 32.0585 | 0.042% | 81.41 | 42.51 |
| 3200 | Nan | 63.755 | Nan | Nan | 83.85 |

Table C.2: Kang's results.(On a laptop with Intel Core i5(4 cores), 1.6GHz, 8GB RAM)

**Comment** : we can notice that, in both tests, the algorithm SFW is way faster than the resolution of GUROBI and the SFW give a solution with a good approximation to the real value. We can therefore conclude that our algorithm is valid and can be used as an approximated solution to solve large-scale problems.

# Bibliography


[1] Kang Liu, Nadia Oudjane, and Laurent Pfeiffer. *Decentralized resolution of finite-state, non-convex, and aggregative optimal control problems*. 2022. arXiv: 2204.07080 [math.OC] (pages 1, 2, 5, 6, 16).

[2] Martin Jaggi. "Revisiting Frank-Wolfe: Projection-free sparse convex optimization". In: *International conference on machine learning*. PMLR. 2013, pp. 427–435 (page 5).

[3] Marguerite Frank and Philip Wolfe. "An algorithm for quadratic programming". In: *Naval Research Logistics Quarterly* 3.1-2 (), pp. 95–110. DOI: https://doi.org/10.1002/nav.3800030109 (page 5).