Université Sidi Mohamed Ben Abdellah de Fès La Faculté des Sciences Dhar El Mahraz

Compte rendu sur l'Utilisation de Sumo pour la Simulation Urbaine et l'Analyse de la pollution Automobile

Réalisé par **Bakir Mohammed**

Master **: Mlaim**

Enseignant: Pr.Noureddine EN-NAHNAHI

Extraction de données et Lancement plusieurs simulation:

```
import os
import subprocess
import xml.etree.ElementTree as ET
import pandas as pd

# Configuration
config_file = "region.sumocfg"
num_runs = 5
sumo_command = "sumo"  # ou "sumo-gui" si tu veux voir la simulation


emission_file = "emissions.xml"
pollution_file = "pollution_output.xml"
queue_file = "queue_detector_output.xml"

# Résultats accumulés
emissions_data = []
pollution_data = []
queue_data = []

def parse_emissions(file, run):
    tree = ET.parse(file)
    root = tree.getroot()
    for timestep in root.findall("timestep"):
        time = timestep.attrib["time"]
        for veh in timestep.findall("vehicle"):
            row = veh.attrib.copy()
            row["run"] = run
            row["time"] = time
            emissions_data.append(row)

def parse_pollution(file, run):
    tree = ET.parse(file)
    root = tree.getroot()
    for interval in root.findall("interval"):
        row = interval.attrib.copy()
        row["run"] = run
        pollution_data.append(row)

def parse_queue(file, run):
    tree = ET.parse(file)
    root = tree.getroot()
```

```
    for interval in root.findall("interval"):
        row = interval.attrib.copy()
        row["run"] = run
        queue_data.append(row)
#Lancer plusieurs simulations
for i in range(num_runs):
    print(f"🔁 Simulation {i+1}/{num_runs}...")
    subprocess.run([sumo_command, "-c", config_file], stdout=subprocess.DEVNULL)

    parse_emissions(emission_file, i + 1)
    parse_pollution(pollution_file, i + 1)
    parse_queue(queue_file, i + 1)

print("✅ Simulations terminées. Création du fichier Excel...")

# Convertir en DataFrames
df_emissions = pd.DataFrame(emissions_data)
df_pollution = pd.DataFrame(pollution_data)
df_queue = pd.DataFrame(queue_data)

# Sauvegarder dans Excel avec plusieurs feuilles
with pd.ExcelWriter("output.xlsx", engine="openpyxl") as writer:
    df_emissions.to_excel(writer, sheet_name="Emissions", index=False)
    df_pollution.to_excel(writer, sheet_name="Pollution", index=False)
    df_queue.to_excel(writer, sheet_name="Queue", index=False)

print("📄 Données stockées dans 'output.xlsx'")
```

Lancer plusieurs simulations SUMO (simulateur de trafic), extraire les données de sortie (émissions, pollution, files d'attente) à chaque exécution, puis **regrouper tous les résultats** dans un **fichier Excel multi-feuilles**.

Visualisation :

Chargement de données :

```
df_emissions = pd.read_excel("D:\\smartCities\\MohammedBakirTutoriel1\\output.xlsx", sheet_name="Emissions")
df_pollution = pd.read_excel("D:\\smartCities\\MohammedBakirTutoriel1\\output.xlsx", sheet_name="Pollution")
df_queue = pd.read_excel("D:\\smartCities\\MohammedBakirTutoriel1\\output.xlsx", sheet_name="Queue")
```

```
df_emissions.head(5)
```

	id	eclass	CO2	CO	HC	NOx	PMx	fuel	electricity	noise	...	type	waiting	lane	pos	speed	angle	x	y	run	time
0	f1.0	HBEFA3/PC_G_EU4	2624.72	164.78	0.81	1.20	0.07	837.22	0.0	55.94	...	car	0.0	edge1_0	5.10	0.00	90.0	5.10	-1.6	1	0.0
1	f1.0	HBEFA3/PC_G_EU4	4097.93	152.03	0.79	1.83	0.09	1307.09	0.0	68.37	...	car	0.0	edge1_0	7.70	2.60	90.0	7.70	-1.6	1	1.0
2	f1.0	HBEFA3/PC_G_EU4	4925.05	137.04	0.74	2.16	0.10	1570.88	0.0	67.52	...	car	0.0	edge1_0	12.49	4.79	90.0	12.49	-1.6	1	2.0
3	f1.0	HBEFA3/PC_G_EU4	4953.24	115.79	0.64	2.13	0.10	1579.87	0.0	66.07	...	car	0.0	edge1_0	18.96	6.47	90.0	18.96	-1.6	1	3.0
4	f1.1	HBEFA3/PC_G_EU4	2624.72	164.78	0.81	1.20	0.07	837.22	0.0	55.94	...	car	0.0	edge1_0	5.10	0.00	90.0	5.10	-1.6	1	3.0

5 rows × 21 columns

```
df_pollution.head(5)
```

	begin	end	id	sampledSeconds	nVehEntered	nVehLeft	nVehSeen	meanSpeed	meanTimeLoss	meanOccupancy	...	meanHaltingDuration	maxHalti
0	0.0	1.0	pollution_detector	0.0	0	0	0	-1.0	-1.0	0.0	...	0.0	
1	1.0	2.0	pollution_detector	0.0	0	0	0	-1.0	-1.0	0.0	...	0.0	
2	2.0	3.0	pollution_detector	0.0	0	0	0	-1.0	-1.0	0.0	...	0.0	
3	3.0	4.0	pollution_detector	0.0	0	0	0	-1.0	-1.0	0.0	...	0.0	
4	4.0	5.0	pollution_detector	0.0	0	0	0	-1.0	-1.0	0.0	...	0.0	

5 rows × 27 columns

```
df_queue.head(5)
```

	begin	end	id	nVehContrib	flow	occupancy	speed	harmonicMeanSpeed	length	nVehEntered	run
0	0.0	1.0	queue_detector_edge1	0	0.0	0.0	-1.0	-1.0	-1.0	0	1
1	1.0	2.0	queue_detector_edge1	0	0.0	0.0	-1.0	-1.0	-1.0	0	1
2	2.0	3.0	queue_detector_edge1	0	0.0	0.0	-1.0	-1.0	-1.0	0	1
3	3.0	4.0	queue_detector_edge1	0	0.0	0.0	-1.0	-1.0	-1.0	0	1
4	4.0	5.0	queue_detector_edge1	0	0.0	0.0	-1.0	-1.0	-1.0	0	1

La corrélation entre la longueur de la fille d'attente et la pollution :

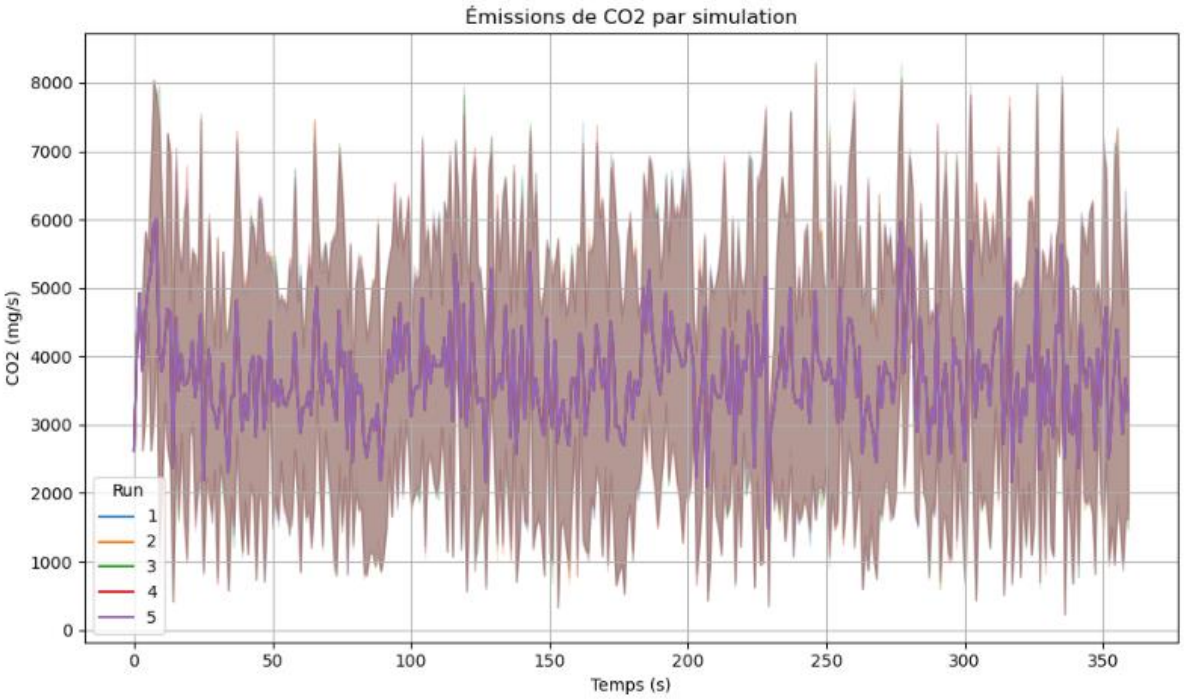
```
# Agréger les émissions de CO2 par time et run (en les sommant par timestep)
df_emissions_agg = df_emissions.groupby(["run", "time"])["CO2"].sum().reset_index()

# Agréger les longueurs de file d'attente par time et run (moyenne ou somme possible)
df_queue_agg = df_queue.groupby(["run", "time"])["length"].mean().reset_index()

# 🩹 Fusionner les deux sur run et time
df_merged = pd.merge(df_emissions_agg, df_queue_agg, on=["run", "time"])
df_merged1 = pd.merge(df_emissions, df_queue, on=["run", "time"])
correlation, _ = pearsonr(df_merged1["length"],df_merged1["CO2"])

print(f"La corrélation est : {correlation:.2f}")
```

```
La corrélation est : 0.03
```

Corrélation **de 0.03 → faible corrélation** indique une très faible relation linéaire entre la longueur des files d'attente et le niveau de pollution (émissions de CO2), **contradiction**

CO2 par timestep pour chaque simulation :

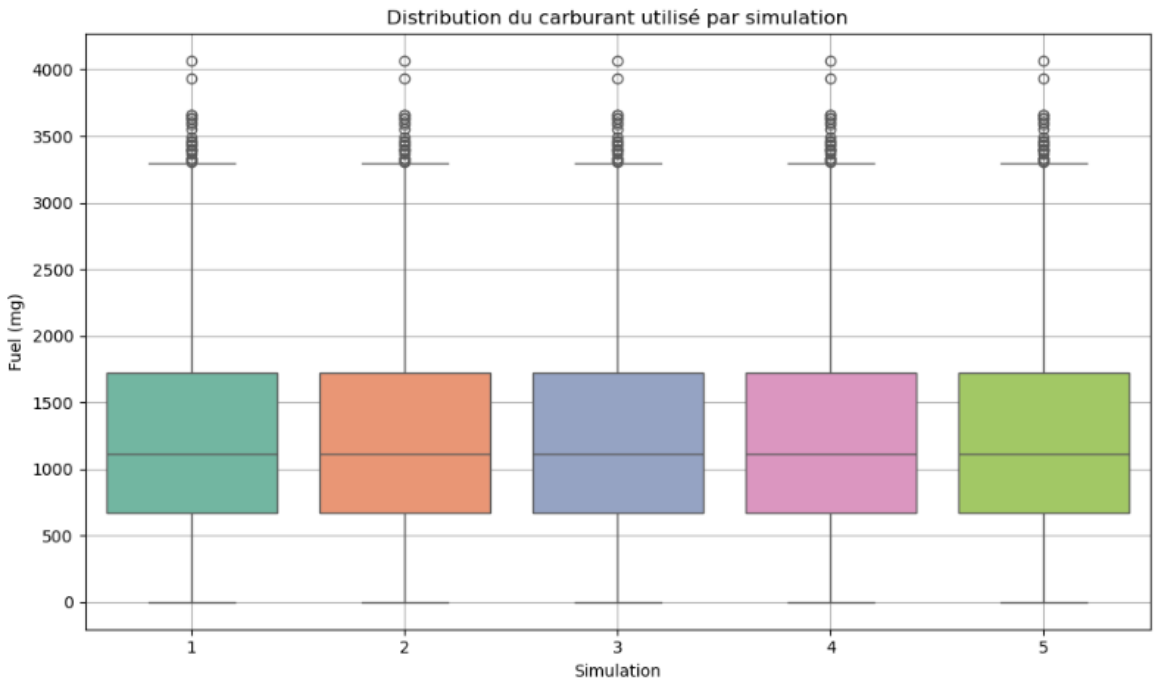
```
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_emissions, x="time", y="CO2", hue="run", palette="tab10")
plt.title("Émissions de CO2 par simulation")
plt.xlabel("Temps (s)")
plt.ylabel("CO2 (mg/s)")
plt.legend(title="Run")
plt.grid(True)
plt.tight_layout()
plt.show()
```



Le système modélisé est **dynamique et chaotique** dans ses émissions de CO2, ce qui est typique du trafic urbain avec des arrêts/redémarrages fréquents. **La moyenne semble stable**, ce qui suggère que le système atteint une forme de régime permanent malgré la variabilité individuelle.

Carburant utilisé par véhicule par simulation :

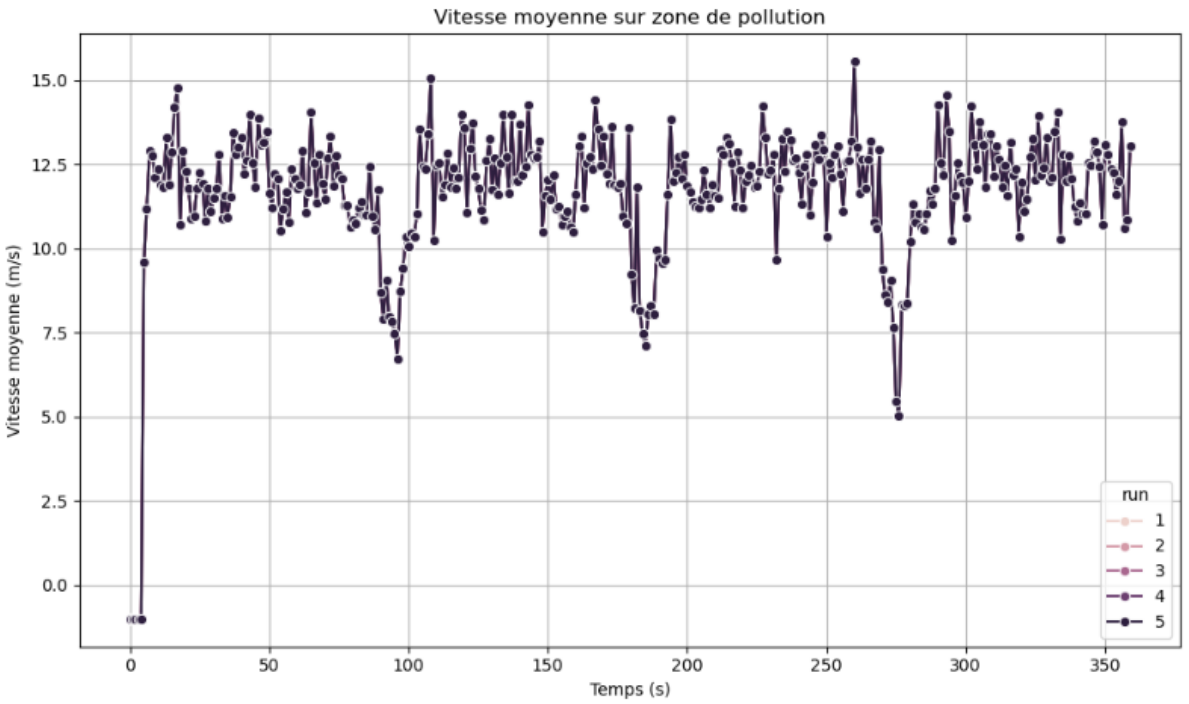
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_emissions, x="run", y="fuel", hue="run", palette="Set2", legend=False)
plt.title("Distribution du carburant utilisé par simulation")
plt.xlabel("Simulation")
plt.ylabel("Fuel (mg)")
plt.grid(True)
plt.tight_layout()
plt.show()
```



Comportement global cohérent entre les différentes simulations : le système est probablement bien contrôlé et les conditions initiales sont similaires.

Pollution: vitesse moyenne par intervalle:

```
df_pollution["meanSpeed"] = pd.to_numeric(df_pollution["meanSpeed"], errors="coerce")
df_pollution["begin"] = pd.to_numeric(df_pollution["begin"], errors="coerce")

plt.figure(figsize=(10, 6))
sns.lineplot(data=df_pollution, x="begin", y="meanSpeed", hue="run", marker="o")
plt.title("Vitesse moyenne sur zone de pollution")
plt.xlabel("Temps (s)")
plt.ylabel("Vitesse moyenne (m/s)")
plt.grid(True)
plt.tight_layout()
plt.show()
```

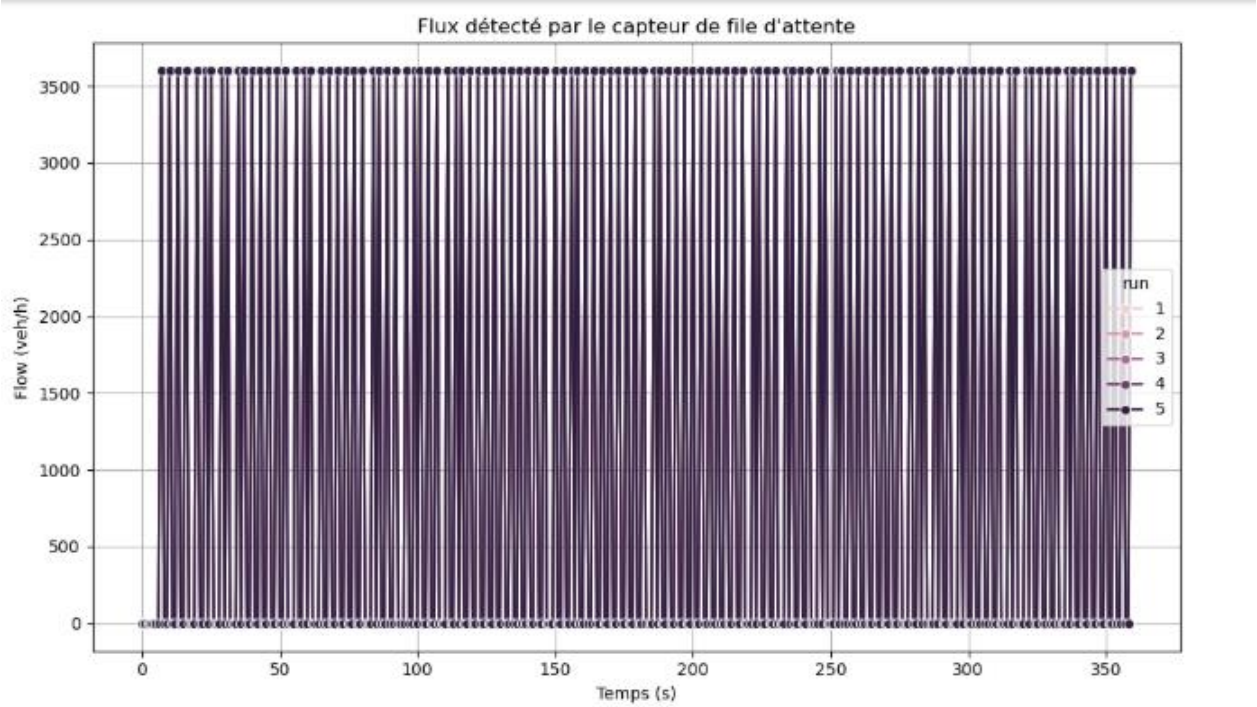


Ce graphique montre l'évolution de la **vitesse moyenne (en m/s)** dans une **zone de pollution** au cours du temps, pour 5 exécutions d'une simulation.

La **vitesse moyenne élevée et stable** pendant une grande partie du temps est un **bon indicateur d'un trafic fluide**, ce qui est en général **positif pour la réduction des émissions**.

Queue detector: flux détecté:

```
df_queue["flow"] = pd.to_numeric(df_queue["flow"], errors="coerce")
df_queue["begin"] = pd.to_numeric(df_queue["begin"], errors="coerce")

plt.figure(figsize=(10, 6))
sns.lineplot(data=df_queue, x="begin", y="flow", hue="run", marker="o")
plt.title("Flux détecté par le capteur de file d'attente")
plt.xlabel("Temps (s)")
plt.ylabel("Flow (veh/h)")
plt.grid(True)
plt.tight_layout()
plt.show()
```



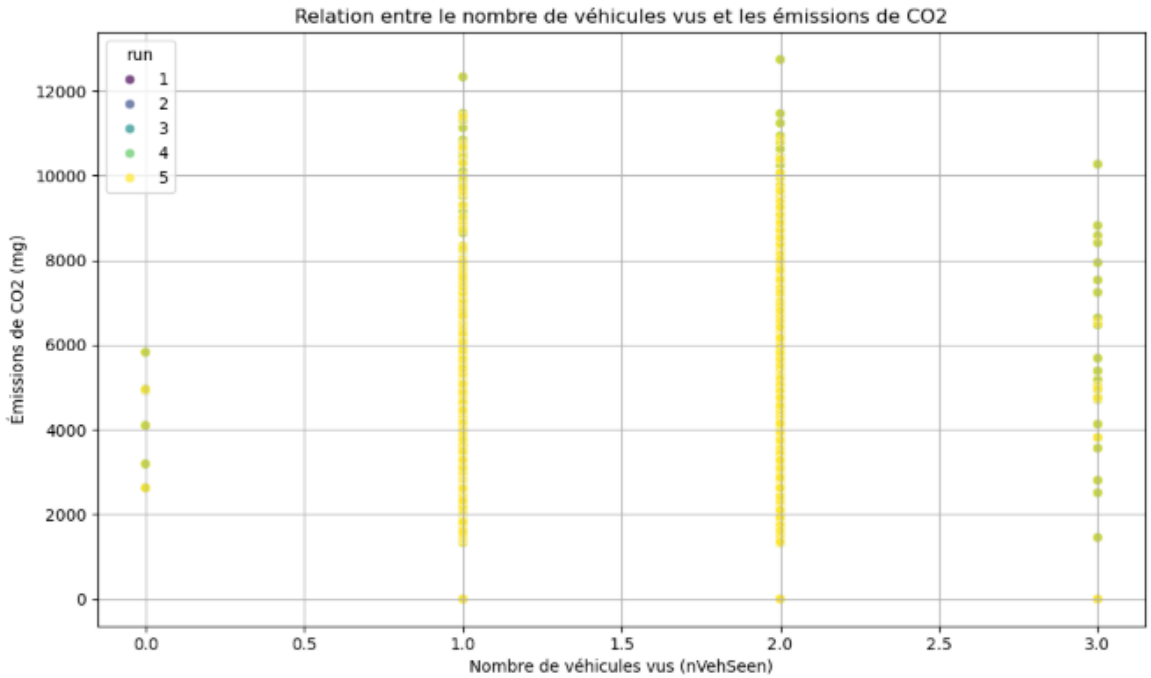
Ce graphique montre l'évolution du **flux détecté par un capteur de file d'attente** (en véhicules par heure, veh/h) en fonction du **temps (en secondes)** pour **5 simulations** différentes.

Relation entre le nombre de véhicules vus et les émissions de CO2 :

#Relation entre le nombre de véhicules vus et les émissions de CO2

17]:
```
df_pollution["time"] = pd.to_numeric(df_pollution["begin"])  # le début de l'intervalle
df_pollution["nVehSeen"] = pd.to_numeric(df_pollution["nVehSeen"], errors="coerce")
df_pollution["run"] = df_pollution["run"].astype(int)

# Fusionner sur time et run
df_merged = pd.merge(df_emissions, df_pollution, on=["time", "run"])
```

18]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_merged, x="nVehSeen", y="CO2", hue="run", palette="viridis", alpha=0.7)
plt.title("Relation entre le nombre de véhicules vus et les émissions de CO2")
plt.xlabel("Nombre de véhicules vus (nVehSeen)")
plt.ylabel("Émissions de CO2 (mg)")
plt.grid(True)
plt.tight_layout()
plt.show()
```



Les résultats montrent une corrélation claire entre le nombre de véhicules vus et les émissions de CO_2. Toutefois, la variabilité des émissions pour un même nombre de véhicules indique que d'autres facteurs influencent les rejets polluants. Les simulations sont cohérentes entre elles, renforçant la fiabilité des observations. Ces résultats suggèrent que la gestion du trafic (notamment la réduction des arrêts/redémarrages) pourrait jouer un rôle clé dans la réduction des émissions.