PROJET CYTECH ING3 - IA

Optimisation Multi-Objectif du Chargement de Véhicules Électriques

Approche par Méta Heuristique (MOPSO) assistée par Modèle de Substitution (Surrogate)

AUTEURS

AIT MOUSSA Amine
DAANOUNI Siham
DELAMOTTE Clément

ENCADREMENT

Mme. Sonia YASSA

Contexte : Le Défi du "Smart Charging"

⚠Problématique Tripartite

La gestion d'une flotte de VE ne se limite pas à la charge. C'est un problème d'optimisation complexe impliquant trois acteurs aux intérêts divergents :

1. Le Gestionnaire (Coût)

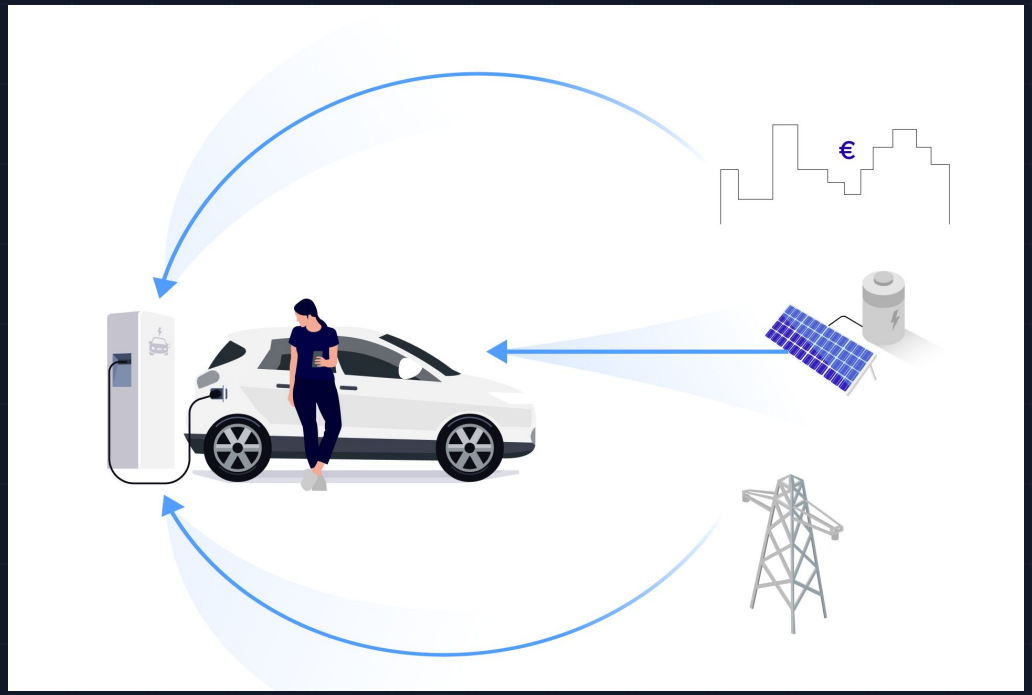
Doit exploiter la volatilité des prix (spot) et utiliser le V2G (décharge rémunérée) pour réduire la facture.

2. L'Utilisateur (Mobilité)

Exige un niveau de charge (SoC) précis à son départ. L'incertitude est inacceptable si l'on veut réduire l'insatisfaction utilisateur.

3. Le Réseau (Infrastructure)

Ne supporte pas toujours les pics de charge simultanés généralement dû aux heures pleines (risque de black-out local).



Interaction complexe entre VE, Réseau et Marché

Modélisation Mathématique : Variables

La Matrice de Décision X

Le problème est discrétisé sur un horizon temporel `T` (ex: 48h) pour `N` véhicules.

$$X \in R^{T \times N}$$

$$x_{i,t} = \text{Puissance du véhicule } i \text{ à l'instant } t$$

- Si `x > 0` : Le véhicule se charge (consomme du réseau).
- Si `x < 0` : Le véhicule décharge (V2G, injecté dans le réseau).

Variables Dérivées

1. État de Charge (SoC) :
Pourcentage de batterie, évolue en fonction de la puissance `x`.

2. Stress Réseau (A_t) :
Somme des puissances à un instant `t`.

$$A_t = \sum_{i=1}^{N} x_{i,t}$$

Fonction Objectif : f_1 Coût Économique

L'objectif est de minimiser le coût total de l'énergie sur l'horizon temporel.

$$f_1(X) = \sum_{t=1}^{T} \varepsilon_t \times A_t \times t$$

ε_t Prix Spot

Tarif de l'électricité à l'instant t (€/kWh). Varie fortement (heures creuses/pleines).

A_t Puissance Nette

Si négative (V2G global), le terme devient un gain financier (revenu).

Stratégie

L'algorithme doit apprendre à charger quand ε_t est bas et décharger quand il est haut.

Fonction Objectif : f_2 Satisfaction Utilisateur

On minimise le déficit de charge au moment du départ (t_{dep}).

$$f_2(X) = \sum_{i=1}^{N} \max 0 , SoC_i^{req} - SoC_{i,t_{dep}}$$

⚠ C'est la fonction la plus coûteuse à calculer car elle dépend de la dynamique temporelle complète.

Dynamique du SoC

Le niveau de batterie à dépend de l'état précédent `t+1` et de la décision `x` :

$$SoC_{i,t+1} = SoC_{i,t} + \frac{x_{i,t} \times t}{C_i}$$

C_i : Capacité totale de la batterie du véhicule `i` .

Fonction Objectif : f_3 Stabilité Réseau

L'objectif est d'éviter les pics de consommation ("Peak Shaving") qui endommagent les transformateurs.

$$f_3(X) = \max_{t \in [1,T]} A_t = \max_t \sum_{i=1}^{N} x_{i,t}$$

Contrainte associée

Contrainte Dure : $\forall t \in [1,T] : A_t \leq A_{\max}$ (Capacité physique du réseau).

Si une solution dépasse A_{max}, elle est réparée par un algorithme glouton (réduction proportionnelle des charges).

Algorithme : MOPSO

Pourquoi MOPSO ?

Multi-Objective Particle Swarm Optimization est idéal pour les problèmes continus.

- Convergence rapide grâce à la mémoire sociale.
- Adapté aux espaces de recherche continus ($x \in R$).

Équation de Vitesse

Le cœur de l'algorithme. La vitesse `v` détermine le déplacement de la particule pour que chacune se rapproche d'un "leader" :

$$v_{i,t}^{new} = w \cdot v + c_1 r_1 \left(P_{best} - x \right) + c_2 r_2 \left(Leader - x \right)$$

```python
def iterate(self):
    if not self.surrogate:
        for t in range(self.t):
            self.select_leader() # Selection of a leader
            for i in range(self.n):
                # Updating velocity and positions
                self.particles[i].update_velocity(leader_pos=self.leader.x, c1=self.c1, c2=self.c2, w=self.w)
                self.particles[i].update_position()

                # Checking boundaries + updating global state
                self.particles[i].keep_boudaries(self.A_max)
                self.particles[i].updating_socs(self.socs, self.capacities)

                # Evaluating particles
                self.particles[i].evaluate(self.prices, self.socs, self.socs_req, self.times)
                self.particles[i].update_best()

            # Update the archive
            self.update_archive()
```

```python
self.n = n # Number of particles
self.t = t # Number of simulation iterations
self.w = w # Inertia (for exploration)
self.c1 = c1 # Individual trust
self.c2 = c2 # Social trust
self.archive_size = archive_size # Archive size

self.surrogate = surrogate # Using AI calculation

# Initialisation of particle's global parameters
self.A_max = A_max # Network's power limit
self.socs, self.socs_req = self.generate_state_of_charges(nb_vehicles,nb_of_ticks)
self.times = self.generate_times(nb_vehicles, nb_of_ticks)
self.prices = self.generates_prices(nb_of_ticks,price_mean,price_std)
self.capacities = capacities
```

Gestion Multi-Objectif : Dominance & Archive

Contrairement à l'optimisation mono-objectif, il n'y a pas une solution unique, mais un ensemble de compromis.

Dominance de Pareto

Une solution `A` domine `B` si :

- A est meilleure que B sur tous les objectif.
- A est strictement meilleure que B sur au moins un objectif.

$$\forall k : f_k(A) \leq f_k(B) \ \wedge \ \exists k : f_k(A) < f_k(B)$$

Archive Externe

MOPSO maintient une "Archive" de taille limitée contenant uniquement les solutions non-dominées trouvées jusqu'ici.

Sélection du Leader : Une particule choisit son guide (Global Best) non pas comme le meilleur absolu, mais en piochant dans cette archive (pour maintenir la diversité). Cela créer une “direction” vers laquelle se déplacer.

Innovation : Le Modèle de Substitution (Surrogate)

Problème : le Goulot d'Étranglement

Pour calculer f_2 (Satisfaction), il faut simuler toute la chronologie de la batterie (boucle pour véhicules).

Complexité : O (T x N)

La Solution : Surrogate

Remplacer ce calcul coûteux par une approximation instantanée via IA, dans notre cas en utilisant MLP Regressor ou Random Forest.

Architecture "Smart MOPSO"

L'algorithme alterne entre calcul exact (pour entraîner l'IA) et prédiction (pour accélérer).

1. Génération

Les particules bougent (MOPSO).

2. Condition

Modèle entraîné

&

Itération % 10 != 0 ?

SI OUI : Prédiction

Utiliser MLP/RF pour estimer f_2 Instantané.

SINON : Calcul Exact

Simulation complète + Ajout au Dataset d'entraînement.

Cadre Expérimental

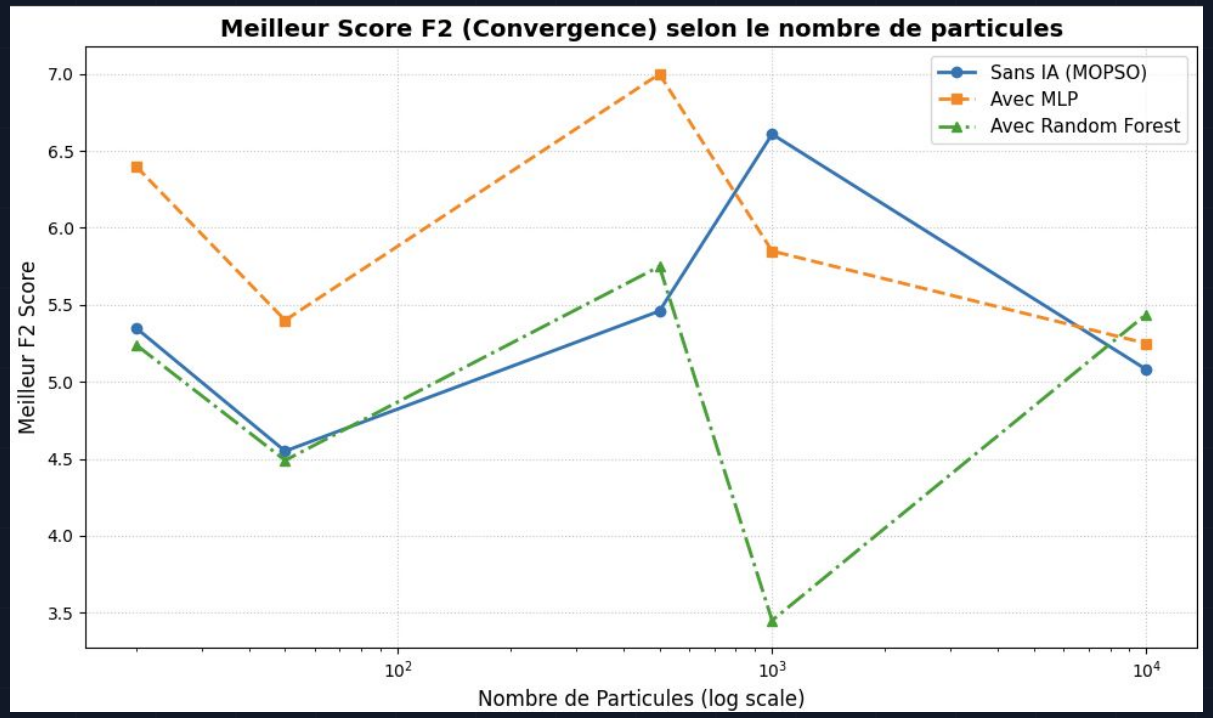
Environnement de Simulation

- **Flotte :** véhicules hétérogènes. `N = 20`
- **Horizon :** (2 jours, avec un pas de 1h). `T = 48`
- **Capacités :** De 11.4 kWh (Hybride) à 200 kWh (Truck).
- **Données Prix :** Réelles (issues de RTE France Eco2Mix).

Paramètres Algorithmiques

Paramètre	Valeur	Rôle
Population	20	Nombre de solutions parallèles
Inertie (w) `w`	0.4	Favorise l'exploitation locale
Cognitif (c1) `c1`	0.3	Confiance en soi
Social (c2) `c2`	0.2	Influence du groupe

Résultats Comparatifs



Interprétation

Le surcoût temporel (entraînement du modèle) est dans tous nos cas dominé par l'approche classique. Cependant, concernant le meilleur résultat de f2 (donc le minimum), le surrogate, dans certains cas, donne de meilleurs résultats, notamment à 1000 particules.

Conclusion et Perspectives Futures

Apports du Projet

- Modélisation mathématique robuste (3 objectifs antagonistes).
- Comparatif explicite de l'hybridation **Métaheuristique et IA (Surrogate)**.
- Utilisation de données réelles (véhicules commerciaux, prix marché).

Perspectives

- **Passage à l'échelle :** Tester sur plus de véhicules. L'avantage du Surrogate devrait exploser car le coût de augmente linéairement avec N. f_2 `N > 100`
- **Incertitudes :** Intégrer des arrivées/départs stochastiques.
- **4ème Objectif :** Minimiser la dégradation de la batterie (cycle aging).