CYTECH

ING 3 - IA

Optimisation Multi-Objectif du Chargement de Véhicules Électriques par MOPSO avec Surrogate

Optimisation Métaheuristique

Auteurs :
AIT MOUSSA Amine
DAANOUNI Siham
DELAMOTTE Clément

Encadrante :
YASSA Sonia
Enseignante-chercheuse en informatique

Année universitaire 2025-2026

Table des matières

Chapitre 1

Contexte et Objectifs du Projet

1.1 Contexte

La transition énergétique mondiale vers une mobilité décarbonée impose des défis considérables aux infrastructures électriques. L'adoption croissante des véhicules électriques (VE) engendre une augmentation significative de la demande en électricité, particulièrement durant les périodes de pointe. Cette situation soulève des problématiques multidimensionnelles touchant à la fois les opérateurs de réseaux électriques, les gestionnaires de flottes et les utilisateurs finaux.

La recharge simultanée et non coordonnée de plusieurs véhicules électriques peut provoquer des pics de consommation susceptibles de déstabiliser le réseau de distribution local. Parallèlement, les fluctuations tarifaires de l'électricité, notamment observées sur le marché spot français (données RTE éco2mix), offrent des opportunités d'optimisation économique. Enfin, la satisfaction des utilisateurs dépend directement de la capacité du système à garantir un niveau de charge suffisant au moment du départ du véhicule.

1.2 Problématique

Le problème étudié consiste à déterminer une stratégie optimale de chargement/-déchargement pour une flotte de véhicules électriques sur un horizon temporel discret. Cette stratégie doit concilier trois objectifs antagonistes :

1. **Minimisation du coût économique (f1)** : réduire la facture électrique globale en exploitant les variations tarifaires
2. **Satisfaction des utilisateurs (f2)** : garantir un état de charge (State of Charge, SoC) conforme aux besoins au moment du départ
3. **Stabilité du réseau (f3)** : limiter la sollicitation maximale du réseau électrique pour éviter les surcharges

Cette problématique relève de l'optimisation multi-objectif sous contraintes, où les solutions ne sont pas uniques mais forment un ensemble de compromis optimaux au sens de Pareto.

1.3 Objectifs du Projet

Les objectifs de ce projet sont les suivants :

1. **Modélisation mathématique** : formaliser le problème d'optimisation de la recharge de VE comme un problème multi-objectif avec trois fonctions antagonistes
2. **Développement algorithmique** : implémenter une métaheuristique Multi-Objective Particle Swarm Optimization (MOPSO) adaptée au problème
3. **Accélération par apprentissage par IA** : intégrer un modèle de substitution (surrogate model) pour réduire le temps de calcul des évaluations coûteuses
4. **Validation expérimentale** : évaluer les performances de l'approche proposée et comparer l'impact de l'utilisation de modèles de substitution

L'approche proposée vise à offrir une solution scalable et efficace pour la gestion intelligente de flottes de véhicules électriques, tout en ouvrant des perspectives d'amélioration par l'hybridation métaheuristique-apprentissage.

Chapitre 2

Modèle Mathématique

2.1 Variables de Décision

Soit un horizon temporel discret divisé en T intervalles (ticks) de durée Δt (exprimée en minutes). On considère une flotte de N véhicules électriques.

2.1.1 Variable Principale

La matrice de décision $\mathbf{X} \in \mathbb{R}^{T \times N}$ où :

— $x_{i,t}$ représente la puissance de charge (si positive) ou de décharge (si négative) du véhicule i à l'instant t, exprimée en kilowatts (kW)

— $x_{i,t} \in [x_{\min}, x_{\max}]$ avec $x_{\min} < 0$ (décharge maximale) et $x_{\max} > 0$ (charge maximale)

2.1.2 Variables Dérivées

— $\mathrm{SoC}_{i,t}$: état de charge du véhicule i à l'instant t, exprimé en pourcentage ($\in [0, 1]$)

— A_t : stress total du réseau à l'instant t, défini par :

$$A_t = \sum_{i=1}^{N} x_{i,t} \tag{2.1}$$

2.2 Paramètres du Problème

Le tableau 2.1 récapitule l'ensemble des paramètres du problème.

2.3 Fonctions Objectif

Le problème d'optimisation multi-objectif vise à minimiser simultanément trois fonctions.

2.3.1 Fonction f_1 : Coût Énergétique Total

Cette fonction calcule le coût total de l'électricité consommée (ou le revenu généré par la décharge) sur l'horizon temporel :

TABLE 2.1 – Paramètres du problème d'optimisation

Symbole	Description	Unité
N	Nombre de véhicules dans la flotte	-
T	Nombre d'intervalles temporels	-
Δt	Pas de temps	min
ε_t	Prix de l'électricité à l'instant t	€/kWh
C_i	Capacité de la batterie du véhicule i	kWh
$\mathrm{SoC}_i^{\mathrm{init}}$	État de charge initial du véhicule i	%
$\mathrm{SoC}_i^{\mathrm{req}}$	État de charge requis du véhicule i	%
t_i^{arr}	Instant d'arrivée du véhicule i	tick
t_i^{dep}	Instant de départ du véhicule i	tick
$A_{\max}$	Puissance maximale admissible par le réseau	kW

$$f_1(\mathbf{X}) = \sum_{t=1}^{T} \varepsilon_t \cdot A_t \cdot \Delta t \tag{2.2}$$

où ε_t est le tarif horaire de l'électricité (€/kWh), $A_t = \sum_{i=1}^{N} x_{i,t}$ est la puissance nette prélevée au réseau, et $\Delta t/60$ assure la conversion en heures.

Cette fonction traduit l'objectif économique : profiter des périodes tarifaires basses pour charger et potentiellement décharger durant les périodes tarifaires élevées (stratégie Vehicle-to-Grid, V2G).

2.3.2 Fonction f_2 : Insatisfaction des Utilisateurs

Cette fonction quantifie le déficit de charge par rapport aux attentes des utilisateurs :

$$f_2(\mathbf{X}) = \sum_{i=1}^{N} \max\left(0, \mathrm{SoC}_i^{\mathrm{req}} - \mathrm{SoC}_{i,t_i^{\mathrm{dep}}}\right) \tag{2.3}$$

où $\mathrm{SoC}_{i,t_i^{\mathrm{dep}}}$ est l'état de charge du véhicule i à son instant de départ.

La fonction $\max(0, \cdot)$ garantit que seuls les déficits de charge sont pénalisés. L'évolution temporelle du SoC est modélisée par la relation de récurrence :

$$\mathrm{SoC}_{i,t+1} = \mathrm{SoC}_{i,t} + \frac{x_{i,t} \cdot \Delta t * 60}{C_i} \tag{2.4}$$

Cette équation traduit la dynamique de charge/décharge, où l'énergie échangée ($x_{i,t} \cdot \Delta t * 60$ en kWh) est convertie en variation de SoC en divisant par la capacité totale de la batterie.

2.3.3 Fonction f_3 : Stress Maximal du Réseau

Cette fonction mesure le pic de sollicitation du réseau électrique :

$$f_3(\mathbf{X}) = \max_{t \in [1,T]} A_t = \max_{t \in [1,T]} \sum_{i=1}^{N} x_{i,t} \tag{2.5}$$

Minimiser cette fonction permet de lisser la courbe de charge et d'éviter les pics de consommation qui sollicitent excessivement les infrastructures de distribution.

2.4 Contraintes

Le problème est soumis à plusieurs contraintes explicites et implicites.

2.4.1 Contraintes Physiques

Contrainte de capacité de la batterie :

$$\forall i \in [1, N], \forall t \in [1, T] : 0 \leq \mathrm{SoC}_{i,t} \leq 1 \tag{2.6}$$

Contrainte de puissance instantanée :

$$\forall i \in [1, N], \forall t \in [1, T] : x_{\min} \leq x_{i,t} \leq x_{\max} \tag{2.7}$$

2.4.2 Contraintes Temporelles

Contrainte de disponibilité :

$$\forall i \in [1, N], \forall t \notin [t_i^{\mathrm{arr}}, t_i^{\mathrm{dep}}[: x_{i,t} = 0 \tag{2.8}$$

Condition initiale :

$$\forall i \in [1, N] : \mathrm{SoC}_{i,t_i^{\mathrm{arr}}} = \mathrm{SoC}_i^{\mathrm{init}} \tag{2.9}$$

2.4.3 Contrainte Réseau

Contrainte de capacité du réseau :

$$\forall t \in [1, T] : A_t \leq A_{\max} \tag{2.10}$$

Cette contrainte est implémentée de manière itérative par un algorithme glouton qui réduit progressivement les puissances de charge (en préservant les puissances de décharge) jusqu'à satisfaction de la contrainte.

2.5 Formulation du Problème Multi-Objectif

Le problème complet s'écrit :

$$\begin{aligned} \min_{\mathbf{X}} \quad & (f_1(\mathbf{X}), f_2(\mathbf{X}), f_3(\mathbf{X})) \\ \text{s.c.} \quad & 0 \leq \mathrm{SoC}_{i,t} \leq 1, \quad \forall i, t \\ & x_{\min} \leq x_{i,t} \leq x_{\max}, \quad \forall i, t \\ & x_{i,t} = 0, \quad \forall i, \forall t \notin [t_i^{\mathrm{arr}}, t_i^{\mathrm{dep}}[\\ & A_t \leq A_{\max}, \quad \forall t \end{aligned} \tag{2.11}$$

Ce problème appartient à la classe des problèmes d'optimisation multi-objectif non linéaires sous contraintes (MONLP). L'existence d'objectifs antagonistes conduit à l'existence d'un ensemble de solutions Pareto-optimales.

Chapitre 3

Approche de Résolution : MOPSO avec Modèle de Substitution

3.1 Justification du Choix de MOPSO

Parmi les métaheuristiques multi-objectifs, l'algorithme **Multi-Objective Particle Swarm Optimization (MOPSO)** a été retenu pour les raisons suivantes :

1. **Efficacité pour les problèmes continus** : les variables $x_{i,t}$ sont des puissances continues, un domaine où PSO excelle
2. **Faible nombre de paramètres** : comparé à NSGA-II ou MOEA/D, MOPSO nécessite moins de réglages
3. **Convergence rapide** : mécanisme de mémoire sociale et individuelle (p_{best})
4. **Adaptabilité** : facilité d'intégration d'un mécanisme d'archive externe

3.2 Principe de MOPSO

3.2.1 Représentation des Particules

Chaque particule représente une solution candidate, définie par :

— **Position x** : matrice $T \times N$ des puissances de charge/décharge

— **Vitesse v** : matrice $T \times N$ représentant la variation de position

— **Meilleure position personnelle $\mathbf{p}_{\text{best}}$** : meilleure position visitée (au sens de Pareto)

— **Meilleure évaluation $\mathbf{f}_{\text{best}}$** : triplet (f_1, f_2, f_3) associé à $\mathbf{p}_{\text{best}}$

3.2.2 Mécanisme de Dominance de Pareto

Une solution A domine une solution B (noté $A \prec B$) si et seulement si :

— A est au moins aussi bonne que B sur tous les objectifs : $f_k(A) \leq f_k(B), \forall k \in \{1, 2, 3\}$

— A est strictement meilleure que B sur au moins un objectif : $\exists k$ tel que $f_k(A) < f_k(B)$

3.2.3 Archive des Solutions Non-Dominées

Un mécanisme d'archive externe conserve les meilleures solutions. L'algorithme de mise à jour est le suivant :

Algorithm 1 Mise à jour de l'archive

```
candidates ← archive ∪ particules
non_dominés ← ∅
for chaque c_i dans candidates do
    dominé ← False
    for chaque c_j dans candidates, i ≠ j do
        if c_j ≺ c_i then
            dominé ← True
            break
        end if
    end for
    if not dominé then
        Ajouter c_i à non_dominés
    end if
end for
if |non_dominés| > archive_size then
    Sélectionner aléatoirement archive_size solutions
end if
archive ← non_dominés
```

3.3 Description Détaillée de l'Algorithme MOPSO

3.3.1 Initialisation

Étape 1 : Génération des paramètres du problème
our chaque instant $t \in [1, T]$:

— Génération d'un prix : $\varepsilon_t = \mu_{\text{price}} + \mathcal{U}(-\sigma_{\text{price}}, \sigma_{\text{price}})$

— Alternative : extraction des prix réels depuis les données RTE France (`elec_prices.csv`)

La génération aléatoire utilise une distribution uniforme pour garantir une couverture complète de l'intervalle $[\mu - \sigma, \mu + \sigma]$, permettant de simuler des variations tarifaires réalistes.

Étape 2 : Initialisation des particules
Pour chaque particule $p \in [1, n]$:

1. Génération de la position initiale : $\forall i, t : x_{i,t}^p \sim \mathcal{U}(x_{\min}, x_{\max})$
2. Application de la contrainte temporelle : $\forall i, \forall t \notin [t_i^{\text{arr}}, t_i^{\text{dep}}[: x_{i,t}^p \leftarrow 0$
3. Génération de la vitesse : $\forall i, t : v_{i,t}^p \sim \mathcal{U}(-\alpha \cdot (x_{\max} - x_{\min}), \alpha \cdot (x_{\max} - x_{\min}))$
4. Évaluation et initialisation de $\mathbf{p}_{\text{best}}^p$

3.3.2 Boucle Principale

Pour chaque itération iter $\in [1, T_{\max}]$:
Étape 1 : Sélection du leader
Le leader est sélectionné aléatoirement parmi l'archive :

$$\text{leader} \leftarrow \text{random\_choice(archive)} \tag{3.1}$$

Étape 2 : Mise à jour des particules (initialisation de l'essaim
Pour chaque particule $p \in [1, n]$:
a) Mise à jour de la vitesse :

$$v_{i,t}^{p,\text{new}} = w \cdot v_{i,t}^{p} + c_1 \cdot r_{1,i} \cdot (p_{\text{best}_{i,t}}^{p} - x_{i,t}^{p}) + c_2 \cdot r_{2,i} \cdot (\text{leader}_{i,t} - x_{i,t}^{p}) \tag{3.2}$$

où :

— w : inertie (exploration vs exploitation)

— c_1 : coefficient cognitif

— c_2 : coefficient social

— $r_{1,i}, r_{2,i}$: facteurs aléatoires individuels

b) Mise à jour de la position :

$$x_{i,t}^{p,\text{new}} = x_{i,t}^{p} + v_{i,t}^{p,\text{new}} \tag{3.3}$$

c) Application des contraintes :

1. Contrainte temporelle : $\forall i, \forall t \notin [t_i^{\text{arr}}, t_i^{\text{dep}}[: x_{i,t}^{p,\text{new}} \leftarrow 0$
2. Contrainte réseau : réduction itérative si $A_t > A_{\max}$

Étape 3 : Mise à jour de l'archive
Appliquer l'algorithme de mise à jour de l'archive.

3.4 Intégration du Modèle de Substitution

3.4.1 Motivation

L'évaluation de f_2 nécessite une simulation complète (complexité $O(T \cdot N)$). Un modèle de substitution permet des prédictions quasi-instantanées.

3.4.2 Choix du Modèle

Deux architectures ont été implémentées :

1. **Multi-Layer Perceptron (MLP)** : architecture (100, 50), activation ReLU, optimiseur Adam
2. **Random Forest (RF)** : 100 arbres, critère MSE, profondeur non limitée

Le Random Forest est privilégié pour sa robustesse et sa capacité à modéliser des relations non linéaires sans normalisation.

3.4.3 Pipeline d'Apprentissage

Collecte des données : avant itération de SmartMOPSO À chaque évaluation exacte, enregistrement de (x_{flat}, f_2)

Prédiction : utilisation de $(x_{\text{flat}}, prediction_freq = 10$, tel que pour chaque 9 prédictions, on effectue 1 calcul réel :

— f_1, f_3 : calculs exacts (coût négligeable)

— f_2 : prédiction par modèle

3.4.4 Justification du Choix de f_2

Analyse des complexités :

— $f_1 : O(TN) \rightarrow$ calcul instantané

— $f_3 : O(TN) \rightarrow$ calcul instantané

— $f_2 : O(T * N) \rightarrow$ coût dominant

De plus, f_2 présente une structure régulière propice à l'apprentissage.

3.4.5 Architecture SmartMOPSO

La classe `SmartMOPSO` étend `MOPSO` en implémentant la logique hybride décrite dans l'algorithme 2.

Algorithm 2 SmartMOPSO - Logique hybride

for iter = 1 to $T_{\max}$ **do**
  for chaque particule p **do**
    Mise à jour position et vitesse
    Application des contraintes
    if (iter (mod prediction_freq) $\neq 0$) $\wedge$ (use_surrogate = True) **then**
      $f_1 \leftarrow$ calcul exact
      $f_3 \leftarrow$ calcul exact
      $f_2 \leftarrow$ prédiction modèle
    else
      $f_1, f_2, f_3 \leftarrow$ calculs exacts
      Ajouter (x, f_2) aux données d'entraînement
    end if
    Mise à jour p_{best} et archive
  end for
end for

Chapitre 4

Expérimentation

4.1 Description de l'Environnement de Simulation

4.1.1 Caractéristiques de la Flotte

La simulation repose sur une flotte hétérogène de $N = 20$ véhicules électriques :

TABLE 4.1 – Caractéristiques de la flotte de véhicules

Paramètre	**Valeur**
Nombre de véhicules (N)	20
Capacité de batterie (C_i)	Variable (11.4 à 200 kWh)
Puissance de charge max ($x_{\max}$)	Calculée dynamiquement
Puissance de décharge max ($x_{\min}$)	$-x_{\max}$
SoC initial ($\text{SoC}_i^{\text{init}}$)	$\mathcal{U}(0, 100)\%$
SoC requis ($\text{SoC}_i^{\text{req}}$)	$\mathcal{U}(\text{SoC}_i^{\text{init}} + 1, 100)\%$

Les capacités de batterie sont extraites aléatoirement depuis le dataset `vehicle_capacity.csv` contenant 102 modèles de véhicules électriques commerciaux. La flotte est donc hétérogène, reflétant la diversité réelle du parc automobile électrique.

4.1.2 Horizon Temporel

L'horizon temporel est divisé en $T = 48$ intervalles de $\Delta t = 60$ minutes, soit une durée totale de 48 heures (2 jours).

Ce choix permet de modéliser :

— Des cycles de charge/décharge sur deux jours complets

— Des variations tarifaires jour/nuit sur plusieurs périodes

— Des patterns de présence réalistes (arrivée en soirée, départ le matin, retour le soir suivant)

4.1.3 Tarification de l'Électricité

Deux approches :

1. **Génération stochastique** : $\varepsilon_t \sim \mathcal{N}(0.08, 0.04)$ €/kWh
2. **Données réelles RTE France** : 250 observations (hiver/été 2025)

4.1.4 Contrainte Réseau

La puissance maximale admissible du réseau est calculée dynamiquement en fonction du nombre de véhicules et des données de consommation réelles du réseau électrique français.

Données de Consommation RTE France

D'après les données RTE France (éco2mix) pour les périodes hiver 2025 (S2-S5) et été 2025 (S29-S32), les consommations nationales observées sont :

TABLE 4.2 – Consommation électrique nationale (France, 2025)

Période	**Maximum**	**Minimum**
Hiver (S2-S5)	87 028 MWh	46 847 MWh
Été (S29-S32)	52 374 MWh	29 819 MWh

Calcul de $A_{\max}$

La puissance maximale du réseau local est calculée par mise à l'échelle de la consommation nationale :

$$\text{Consommation moyenne} = \frac{87028 + 46847 + 52374 + 29819}{4} = 54017 \text{ MWh} \tag{4.1}$$

$$\text{Ratio simulation} = \frac{N_{\text{véhicules}}}{N_{\text{consommateurs\_France}}} = \frac{20}{67 \times 10^6} \approx 2.985 \times 10^{-7} \tag{4.2}$$

$$A_{\max} = \text{Ratio} \times \text{Consommation moyenne} \approx 0.0161 \text{ kW} \tag{4.3}$$

$$x_{\max} = \frac{A_{\max}}{N} \approx 0.0008 \text{ kW par véhicule} \tag{4.4}$$

$$x_{\min} = -x_{\max} \tag{4.5}$$

Cette approche garantit une contrainte réseau cohérente avec la proportion réelle de véhicules électriques simulés par rapport à la population française.

4.1.5 Environnement Logiciel

— **Langage** : Python 3.11+
— **Bibliothèques** : pandas 2.3.3, numpy 2.4.1, scikit-learn
— **Gestionnaire de paquets** : uv

TABLE 4.3 – Paramètres de l'algorithme MOPSO

Paramètre	**Notation**	**Valeur**	**Justification**
Taille population	n	20	Compromis exploration/temps
Nombre d'itérations	$T_{\max}$	30	Convergence empirique
Inertie	w	0.4	Exploitation ($w < 0.5$)
Coeff. cognitif	c_1	0.3	Réduction confiance individuelle
Coeff. social	c_2	0.2	Équilibre exploration/exploitation
Taille archive	archive_size	10	Diversité du front
Facteur vitesse	α	0.1	Évite oscillations

4.2 Paramétrage de la Métaheuristique

4.2.1 Paramètres MOPSO

Note sur les coefficients c_1 et c_2 : Contrairement aux valeurs classiques de PSO ($c_1 = c_2 = 2.0$), nous avons opté pour des valeurs réduites ($c_1 = 0.3$, $c_2 = 0.2$) afin de favoriser une exploration plus progressive de l'espace de recherche et d'éviter une convergence prématurée vers des optima locaux. Cette configuration s'avère particulièrement adaptée aux problèmes multi-objectifs où la diversité du front de Pareto est cruciale.

4.2.2 Paramètres du Modèle de Substitution

Multi-Layer Perceptron (MLP) :

— Architecture : 2 couches cachées (100, 50 neurones)
— Activation : ReLU
— Époques max : 500
— Optimiseur : Adam

Random Forest (RF) :

— Nombre d'arbres : 100
— Critère : MSE
— Profondeur : illimitée

Paramètres d'apprentissage :

— Fréquence de réentraînement : 10 itérations
— Minimum de données : 20 échantillons

4.3 Protocole de Tests

4.3.1 Scénarios Évalués

Trois configurations testées :

1. **No-AI (baseline)** : MOPSO standard sans modèle de substitution
2. **MLP** : MOPSO avec modèle de substitution MLP
3. **RF** : MOPSO avec modèle de substitution Random Forest

4.3.2 Métriques d'Évaluation

1. **Temps de calcul total** (secondes) : efficacité computationnelle
2. **Qualité de la meilleure solution** : valeur minimale de f_2 dans l'archive

4.3.3 Questions de Recherche

1. Quel est le gain en temps de calcul avec le modèle de substitution ?
2. L'approximation de f_2 détériore-t-elle la qualité des solutions ?
3. Quel modèle (MLP vs RF) offre le meilleur compromis vitesse/précision ?

4.3.4 Limitations

— Absence de runs multiples (pas de moyenne/écart-type)
— Pas de visualisation du front de Pareto
— Métriques de qualité (hypervolume, spacing) non calculées
— Pas d'étude de sensibilité des paramètres
— Pas de comparaison avec d'autres métaheuristiques

4.4 Comparaison avec d'Autres Méthodes

4.4.1 Positionnement de MOPSO

Comparaison avec NSGA-II :
— **NSGA-II** : front bien distribué, mais complexité $O(MN^2)$
— **MOPSO** : convergence plus rapide, mémoire sociale efficace
Comparaison avec MOEA/D :
— **MOEA/D** : décomposition en sous-problèmes, efficace pour nombreux objectifs
— **MOPSO** : évite scalarisation explicite, plus flexible
Comparaison avec approches gloutonnes :
— **Gloutonnes** : rapides mais sous-optimales
— **MOPSO** : exploration globale de l'espace de Pareto

4.4.2 Positionnement du Modèle de Substitution

Comparaison avec parallélisation :
— **Parallélisation** : accélération linéaire, pas de perte de précision
— **Surrogate** : gain sans infrastructure parallèle
Comparaison avec Kriging/GP :
— **Kriging** : estimation d'incertitude, mais coûteux
— **MLP/RF** : plus scalables, entraînement rapide

Chapitre 5

Conclusion et Perspectives

5.1 Synthèse des Contributions

Ce projet a permis de développer une approche complète pour l'optimisation multi-objectif du chargement de véhicules électriques :

1. **Modélisation rigoureuse** : formalisation mathématique avec trois objectifs antagonistes
2. **Implémentation MOPSO** : métaheuristique avec archive externe pour le front de Pareto
3. **Hybridation avec l'utilisation de l'IA** : RF et MLP modèles de substitution pour accélération sans dégradation
4. **Architecture modulaire** : code structuré et réutilisable
5. **Données réelles** : prix RTE France et capacités de batteries commerciales

5.2 Limites Identifiées

5.2.1 Limites Méthodologiques

— Pas d'étude de sensibilité systématique

— Absence de validation croisée du surrogate

— Métriques de qualité du front non calculées

— Runs uniques sans répétitions

5.2.2 Limites Computationnelles

— Scalabilité non testée ($N > 10$, $T > 72$)

— Contraintes simplifiées (pas de dégradation, rampe)

— Résolution temporelle fixe ($\Delta t = 60$ min)

5.2.3 Limites du Modèle

— Flotte homogène

— V2G sans contraintes de dégradation

— Modèle réseau simplifié

5.3 Perspectives d'Amélioration

5.3.1 Extensions Méthodologiques

Validation expérimentale approfondie :

— Runs multiples (30-50) avec statistiques
— Calcul de métriques standards (hypervolume, epsilon-indicator)
— Tests statistiques (Wilcoxon, Kruskal-Wallis)
— Visualisation des fronts de Pareto

Optimisation des hyperparamètres :

— Étude de sensibilité sur w, c_1, c_2
— Meta-learning ou grid search
— Adaptation dynamique des paramètres

Amélioration du surrogate :

— Autres architectures (CNN pour structure 2D, LSTM pour temporalité)
— Uncertainty quantification
— Fréquence de réentraînement adaptative

Comparaisons rigoureuses :

— Implémentation de NSGA-II, MOEA/D
— Hybridations (MOPSO + opérateurs génétiques)

5.3.2 Extensions Fonctionnelles

Modélisation avancée :

— Dégradation batterie (SoH comme 4ème objectif)
— Pertes en ligne et congestion locale
— Contraintes de rampe
— Sources renouvelables avec incertitudes

Gestion d'incertitudes :

— Distributions probabilistes (arrivée/départ)
— Scénarios tarifaires multiples
— Optimisation robuste/stochastique

Scalabilité :

— Parallélisation (OpenMP, multiprocessing)
— Implémentation distribuée ($N > 100$)
— Approches hiérarchiques

5.3.3 Applications Industrielles

Cas d'usage :

— Gestion de flottes (taxis, livraison, transport public)

— Parkings intelligents avec V2G

— Bornes rapides sur autoroutes

— Agrégation de flexibilité pour marchés d'ajustement

Partenariats envisageables :

— Opérateurs réseaux (Enedis, RTE)

— Constructeurs automobiles

— Agrégateurs d'énergie

5.4 Conclusion Générale

Ce projet a démontré la faisabilité et l'intérêt d'une approche hybride couplant métaheuristiques multi-objectifs et apprentissage automatique pour l'optimisation du chargement de véhicules électriques. L'algorithme MOPSO, combiné à un modèle de substitution, offre un compromis prometteur entre qualité des solutions et efficacité computationnelle.

Les résultats obtenus constituent une base solide pour des développements futurs visant à intégrer cette approche dans des systèmes opérationnels de gestion de flottes. L'extensibilité de l'architecture logicielle développée facilite l'intégration de nouvelles fonctionnalités et l'adaptation à des contextes applicatifs variés.

Au-delà des aspects techniques, ce projet illustre l'importance de l'optimisation multi-objectif dans les problématiques énergétiques contemporaines, où la recherche de compromis entre objectifs antagonistes (économie, environnement, confort) est devenue incontournable. L'intégration de l'intelligence artificielle dans les métaheuristiques ouvre des perspectives stimulantes pour la résolution de problèmes d'optimisation complexes à grande échelle, contribuant ainsi à la transition énergétique et à l'émergence de réseaux électriques intelligents.

Bibliographie

[1] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). *A fast and elitist multiobjective genetic algorithm : NSGA-II*. IEEE Transactions on Evolutionary Computation, 6(2), 182-197.

[2] Coello Coello, C. A., Lamont, G. B., & Van Veldhuizen, D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems* (2nd ed.). Springer.

[3] Kennedy, J., & Eberhart, R. (1995). *Particle swarm optimization*. Proceedings of IEEE International Conference on Neural Networks, 4, 1942-1948.

[4] Coello Coello, C. A., & Lechuga, M. S. (2002). *MOPSO : A proposal for multiple objective particle swarm optimization*. Proceedings of Congress on Evolutionary Computation, 2, 1051-1056.

[5] Jin, Y. (2011). *Surrogate-assisted evolutionary computation : Recent advances and future challenges*. Swarm and Evolutionary Computation, 1(2), 61-70.

[6] Forrester, A. I., & Keane, A. J. (2009). *Recent advances in surrogate-based optimization*. Progress in Aerospace Sciences, 45(1-3), 50-79.

[7] Schuller, A., Flath, C. M., & Gottwalt, S. (2015). *Quantifying load flexibility of electric vehicles for renewable energy integration*. Applied Energy, 151, 335-344.

[8] Sortomme, E., & El-Sharkawi, M. A. (2012). *Optimal scheduling of vehicle-to-grid energy and ancillary services*. IEEE Transactions on Smart Grid, 3(1), 351-359.

[9] Qian, J., Jiang, Y., Liu, X., Wang, Q., Wang, T., Shi, Y., & Chen, W. (2023). *Federated Reinforcement Learning for Electric Vehicles Charging Control on Distribution Networks*.https://arxiv.org/pdf/2308.08792

[10] RTE France. (2025). *éco2mix - Données temps réel du système électrique*. Disponible sur : https://www.rte-france.com/donnees-publications/eco2mix-donnees-temps-reel/donnees-marche

[11] Kaggle. (2025). *Car Dataset (2025)*. Disponible sur : https://www.kaggle.com/datasets/abdulmalik1518/cars-datasets-2025/data

Annexe A

Structure du Code Source

```
optim-meta/
          main.py                  # Point d'entree, tests
   comparatifs
          mopso.py                 # Impl mentation MOPSO
          particle.py              # Classe Particle
          surrogate_handler.py     # Modeles de substitution (IA)
          pyproject.toml           # Configuration projet
          uv.lock                  # Dependances figees
          README.md                # Documentation
          data/
              vehicle_capacity.csv   # 102 mod les VE
              elec_prices.csv        # 250 prix RTE
              grid_capacity.txt
```

Listing A.1 – Arborescence du projet

Annexe B

Commandes d'Exécution

B.1 Installation de l'Environnement

```
# Installation de UV (linux)
curl -LsSf https://astral.sh/uv/install.sh | sh

# Installation de UV (Windows)
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/
    install.ps1 | iex"
# Ou via WinGet
winget install --id=astral-sh.uv  -e
```

Listing B.1 – Installation Linux/Windows

```
# Creation d'un environnement virtuel (necessaire)
uv venv

# Telechargement des requirements du projet
uv pip sync uv.lock

# Si uv.lock ne fonctionne pas correctement ou n existe pas,
    vous pouvez le geneer avec la commande suivante a partir du .
    toml:
uv pip compile --upgrade pyproject.toml -o uv.lock
```

Listing B.2 – Installation / mise en place de l'environnement

B.2 Exécution du Programme

```
# Ex cution
uv run main.py
```

Listing B.3 – Lancement du programme principal

Annexe C

Exemple de Sortie Console

```
--- Launching Scenario: Only MOPSO ---
Finished in 9.62 seconds.
Best f2 found: 5.3396

--- Launching Scenario: With MLP ---
Finished in 8.16 seconds.
Best f2 found: 6.0898

--- Launching Scenario: With Random Forest ---
Finished in 69.95 seconds.
Best f2 found: 7.1100

=== SUMMARY ===
Mode             | Time (s)    | Best f2
-----------------------------------------------
No-AI            | 9.62        | 5.3396
MLP              | 8.16        | 6.0898
RF               | 69.95       | 7.1100
```

Listing C.1 – Sortie typique du programme

```

=== SUMMARY ===
                 Mode             | Time (s)    | Best f2
-----------------------------------------------
MOPSO           _20               | 0.39        | 5.6198
MOPSO           _50               | 0.92        | 4.0497
MOPSO           _100              | 1.90        | 6.0193
MOPSO           _500              | 9.62        | 5.3396
MLP             _20               | 0.32        | 6.7202
MLP             _50               | 0.78        | 4.2297
MLP             _100              | 1.80        | 3.2313
MLP             _500              | 8.16        | 6.0898
RF              _20               | 2.77        | 5.7399
RF              _50               | 6.91        | 5.2098
RF              _100              | 13.86       | 5.1697
RF              _500              | 69.95       | 7.1100
```

Listing C.2 – Sortie typique du programme

Annexe D

Extraits de Code Significatifs

D.1 Classe Particle - Fonction Objectif f2

```
def f2(self,socs,socs_req,times):
        result = 0
        for i in range(self.nb_vehicles):
            leaving = times[i][1]
            stress = socs_req[i] - socs[leaving][i]
            result += max(0, stress)
        return result
```

Listing D.1 – Implémentation de f2

D.2 Classe MOPSO - Dominance de Pareto

```
    # True if a dominates b, else false
    def dominates(self, a:Particle, b:Particle):
        dominates = (a.f_current[0] <= b.f_current[0]) and (a.
           f_current[1] <= b.f_current[1]) and (a.f_current[2] <=
            b.f_current[2])
        if dominates:
            # Not strict superiority yet
            dominates = (a.f_current[0] < b.f_current[0]) or (a.
               f_current[1] < b.f_current[1]) or (a.f_current[2]
               < b.f_current[2])
        return dominates
```

Listing D.2 – Relation de dominance

D.3 SmartMOPSO - Utilisation du Surrogate

```
def iterate(self, prediction_freq:int=10):
        # Main loop (overriding original logic to manage control flow)
        for t in range(self.t):
            self.select_leader()

            for i in range(self.n):
                # Movement
                self.particles[i].update_velocity(self.leader.x, self.c1, self.c2, self.w)
                self.particles[i].update_position()
                self.particles[i].keep_boudaries(self.A_max)

                if (t % (prediction_freq) != 0) and self.use_surrogate:
                    # Fast exact calculation (f1, f3)
                    f1 = self.particles[i].f1(self.prices)
                    f3 = self.particles[i].f3()

                    # Slow prediction (f2) by using Surrogate
                    f2_pred = self.surrogate_handler.predict(self.particles[i].x)

                    # Inject scores without running the expensive 'updating_socs'
                    self.particles[i].f_current = [f1, f2_pred, f3]

                else:
                    # Standard Calculation (Slow and Exact)
                    self.particles[i].updating_socs(self.socs, self.capacities)
                    self.particles[i].evaluate(self.prices, self.socs, self.socs_req, self.times)

                self.particles[i].update_best()

            self.update_archive()
```

Listing D.3 – Utilisation du surrogate