Optimisation de tournées de drones à l'aide d'un GNN–PPO

Projet Drone Delivery Optimizer

Mahouna ___ (RTX 4050 / Intel i7)

May 16, 2025

Contents

1 Introduction

Ce rapport présente la modélisation mathématique et l'implémentation d'une méthode hybride pour l'optimisation de tournées de drones :

1. un *Algorithme Génétique* (GA) classique corrigé pour garantir les points de départ et d'arrivée,
2. une approche **PPO** (Proximal Policy Optimization) alimentée par un **GNN** pour l'apprentissage par renforcement.

2 Modèle de réseau de livraison

2.1 Génération des nœuds

On génère N nœuds répartis par Poisson–disc à l'intérieur du polygone *France*, issus du fichier `metropole-version-simplifiee.geojson`.

2.2 Graphe orienté des k-plus-proches-voisins

Chaque nœud u est connecté vers ses $k = 10$ plus proches voisins *sortants*, formant un graphe orienté

$$G_t = (\mathcal{V}_t, \mathcal{E}_t), \quad \mathcal{E}_t \subset \{\, u \to v \,\}.$$

Chaque arête orientée $e = (u \to v)$ porte un coût $c_{uv} = d_{uv} \cdot \Big(\underbrace{1}_{\text{base}} \; \underbrace{-\alpha \cdot w_{uv}}_{\text{effet du vent}} \; \underbrace{+\beta \cdot \eta_{uv}}_{\text{bruit aléatoire}}\Big) \in [0, 600]$, où en général $c_{uv} \neq c_{vu}$.

2.3 Attributs nodaux X

Pour chaque sommet v, on définit

$$x_v = \big[\; \underbrace{\text{one-hot(type)}}_{\in\{\text{hub,pickup,delivery,charging}\}\,(4)}, \; \underbrace{\text{stock/demande}}_{\in\mathbb{R}\,(1)}, \; \underbrace{\lambda_v, \phi_v}_{\text{latitude/longitude}\,(2)}, \; \underbrace{\cos\omega_v, \sin\omega_v}_{\text{direction du vent}\,(2)} \;\big] \in \mathbb{R}^9.$$

Le terme « stock/demande » reste nul ou constant ici car on part d'une hypothèse de demande unitaire illimitée.

3 Fonction de coût généralisée

3.1 Consommation de batterie par arête

Pour une arête e_i de coût c_i, la batterie consommée est

$$\Delta b_i = \frac{c_i}{k}\,(1 + \alpha\,(p_i - 1)), \quad \underbrace{k = 10.8}_{\text{normalisation}}, \quad \underbrace{\alpha = 0.2}_{\substack{\text{facteur de surcharge} \\ \text{(colis multiple)}}}, \quad \underbrace{p_i}_{\text{nombre de colis embarqués}}.$$

Plus p_i est grand, plus la consommation augmente.

3.2 Objectif global d'une tournée E

Pour une séquence d'arêtes $E = (e_1, \ldots, e_T)$, on définit

$$J(E) = \sum_{e_i \in E} c_i \; + \; \lambda \sum_{S} \Big[\max(0, \sum_{e_i \in S} \Delta b_i - B_{\max})\Big]^2 \; + \; \mu \, \#\{\text{recharges}\},$$

où

- $B_{\max} = 100$ est la capacité maximale de batterie,
- $\lambda \gg 1$ pénalise fortement tout dépassement de batterie,
- $\mu \ll \lambda$ pénalise légèrement chaque recharge,
- S parcourt chaque segment consécutif entre deux recharges.

4 Contraintes de début et fin de tournée

4.1 1. Algorithme génétique (GA)

But Générer des chromosomes (tours) garantissant la séquence :

$$H_{\text{start}} \; \to \; D \; \to \; L \; \to \; H_{\text{end}},$$

avec $H.$ hubs et D, L points pickup/delivery.

Initialisation Pour chaque individu :

1. Choix aléatoire d'un hub H_{start}.
2. Construction de shortest_path($H_{\text{start}} \to D$).
3. Ajout de shortest_path($D \to L$).
4. Ajout de shortest_path($L \to H_{\text{end}}$) avec H_{end} choisi aléatoirement.

Le chromosome est la concaténation, en omettant les doublons consécutifs.

Réparation (repair) Après crossover/mutation :

- On repère les indices de D et L.
- On sectionne pour forcer $\cdots \to D \to L \to \ldots$.
- Si la fin n'est pas un hub, on y greffe shortest_path(dernier $\to H_{\text{rand}}$).

Mutation spécifique Échanger deux sous-chemins internes tout en maintenant la séquence $H \to D \to L \to H$.

4.2 2. PPO + GNN (RL)

Intégration dans l'environnement Au sein de l'environnement de RL :

- *État initial virtuel* : l'agent reçoit un état "départ" non relié.
- *Pas 0* : action spéciale

$$a_0 = (\text{téléportation vers un hub } H_i), \quad \text{coût} = c_{\text{pickup}\to H_i}.$$

  Ce hub H_i est tiré aléatoirement parmi ceux $\le d_{\max}$ du point de pickup.
- *Fin de tournée* : après desserte de D et L, l'agent choisit

$$a_T = (\text{téléportation vers hub } H_j), \quad \text{coût} = c_{v_T\to H_j}.$$

- *Termination* : l'épisode se termine dès que tous les flags $\{\text{picked}, \text{delivered}\}$ sont à 1 et qu'un hub est rejoint.

Récompense instantanée mise à jour Pour tout pas t,

$$r_t = -\,c_t - \lambda\left[\max(0, b_t - \Delta b_t)\right]^2 - \mu\,\mathbb{1}_{\{\text{recharge}\}} - \kappa\,\mathbb{1}_{\{\text{téléportation non valide}\}},$$

où $\kappa \gg 1$ pénalise tout téléport illégal hors hub.

5 Formulation RL et Architecture du GNN

5.1 État $s_t = (A_t, X_t, b_t, p_t, v_t)$

5.2 Action $a_t \in \{1, \dots, K\}$

Choix d'une arête orientée sortante ou d'une action spéciale de téléportation.

5.3 GNN – Message Passing

$$H^{(\ell+1)} = \sigma\big(W_1^{(\ell)} H^{(\ell)} + W_2^{(\ell)} A_t\, H^{(\ell)}\big), \quad H^{(0)} = X_t.$$

5.4 Readout global et tête acteur–critique

$$g_t = \frac{1}{|\mathcal{V}_t|}\sum_v H_v^{(L)}, \quad \pi_\theta(a_t \mid s_t) = \text{Softmax}(W_\pi g_t + U_\pi[b_t, p_t]), \quad V_\psi(s_t) = w_v^\top g_t + u_v^\top[b_t, p_t].$$

6 Algorithme PPO

6.1 Avantage (GAE-λ)

$$\hat{A}_t = \sum_{k\ge 0} (\gamma\lambda)^k \big(r_{t+k} + \gamma V_\psi(s_{t+k+1}) - V_\psi(s_{t+k})\big).$$

6.2 Perte « clipped »

$$\mathcal{L} = \mathbb{E}_t\Big[\min\big(r_t\hat{A}_t,\ \text{clip}(r_t, 1\pm\varepsilon)\hat{A}_t\big)\Big] + c_1\|R_t - V_\psi\|^2 - c_2\,\mathcal{H}[\pi_\theta].$$

7 Protocole d'entraînement

- $N = 8$ environnements parallèles.
- $\gamma = 0.99,\ \lambda_{\text{GAE}} = 0.95,\ \varepsilon = 0.2$.
- GNN : 2 couches, dimension 128, ReLU, dropout 0.1.
- Adam, LR 3×10^{-4}, FP16 sur RTX 4050.

8 Conclusion

L'intégration explicite des contraintes de téléportation via un hub garantit la validité des tournées tant dans GA (par génération/réparation) que dans PPO+GNN (par encapsulation dans l'environnement). Le GNN–PPO exploite les coûts orientés c_{uv} et converge plus rapidement qu'un GA classique tout en respectant les contraintes de batterie grâce aux pénalités λ et μ.