# Optimisation de trajectoires robotiques dans un entrepôt Amazon simulé avec OR-Tools et Machine Learning

**Inès Gbadamassi**
ines.gbadamassi@edu.univ-paris13.fr

**Simulation d'entrepôt · OR-Tools · TSP/VRP · ML · Optimisation Hybride**

## Résumé

Ce rapport présente la conception, l'implémentation et l'analyse d'un système complet de planification de trajets robotiques dans un entrepôt simulé. L'objectif est de modéliser un environnement similaire aux infrastructures Amazon Robotics, d'y appliquer des méthodes d'optimisation (TSP/VRP via OR-Tools), et d'intégrer un module de machine learning permettant de prédire les temps de trajet selon la congestion et la présence d'obstacles.

Ce projet personnel illustre une approche hybride combinant simulation géométrique, optimisation déterministe et modélisation statistique.

## 1 Introduction

La robotisation d'entrepôts logistiques constitue un enjeu stratégique majeur pour des acteurs tels qu'Amazon Robotics.

La qualité de la planification des trajets influence directement le débit opérationnel, la consommation énergétique et la fluidité du transport interne. Dans un environnement dense, les robots doivent être capables de contourner la congestion, d'éviter les zones saturées et d'adapter leurs trajectoires.

Dans ce projet, nous développons :
- une simulation d'entrepôt 2D (20×20),
- un solveur TSP/VRP utilisant OR-Tools,
- un générateur de données synthétiques (5000+ échantillons),
- un modèle ML prédisant les temps de trajet,
- une intégration ML + OR permettant d'ajuster les coûts de déplacement.

## 2 Simulation du dépôt

La modélisation de l'entrepôt constitue la première composante essentielle du pipeline. Elle doit fournir un environnement suffisamment riche pour refléter la diversité des scénarios rencontrés dans un entrepôt robotisé réel, tout en restant contrôlable pour l'expérimentation. Notre objectif est d'émuler un cadre simplifié mais cohérent avec les systèmes de transport internes de plateformes logistiques telles que celles d'Amazon Robotics.

### 2.1 Structure spatiale et hypothèses

Nous représentons l'entrepôt sous la forme d'un plan continu de dimensions $20 \times 20$. Ce choix permet d'articuler deux avantages : une structure géométrique simple, facilitant l'analyse, et une flexibilité suffisante pour générer des configurations variées.

Trois types d'entités sont placés aléatoirement :
- **robots** (points de départ des tournées),
- **points de collecte** (où les robots récupèrent des marchandises),
- **stations de livraison**.

Afin d'éviter les ambiguïtés, toutes les positions sont assurées distinctes. Nous utilisons une distribution uniforme sur l'espace continu, évitant ainsi les artefacts réguliers qui pourraient biaiser l'optimisation des trajectoires.

### 2.2 Calcul de la matrice de distances

Une fois les points générés, nous construisons une matrice complète de distances $D \in \mathbb{R}^{n \times n}$ entre tous les nœuds. Deux métriques sont supportées :

$$D_{ij} = \begin{cases} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, & \text{distance euclidienne,} \\ |x_i - x_j| + |y_i - y_j|, & \text{distance Manhattan.} \end{cases} \tag{1}$$

La cohérence de la matrice est vérifiée automatiquement :
- $D_{ii} = 0$,
- $D_{ij} > 0$ pour $i \neq j$,
- $D_{ij} = D_{ji}$ (symétrie),
- absence de valeurs aberrantes.

Une visualisation de l'entrepôt est fournie Fig. 1. Elle valide la distribution spatiale des nœuds.

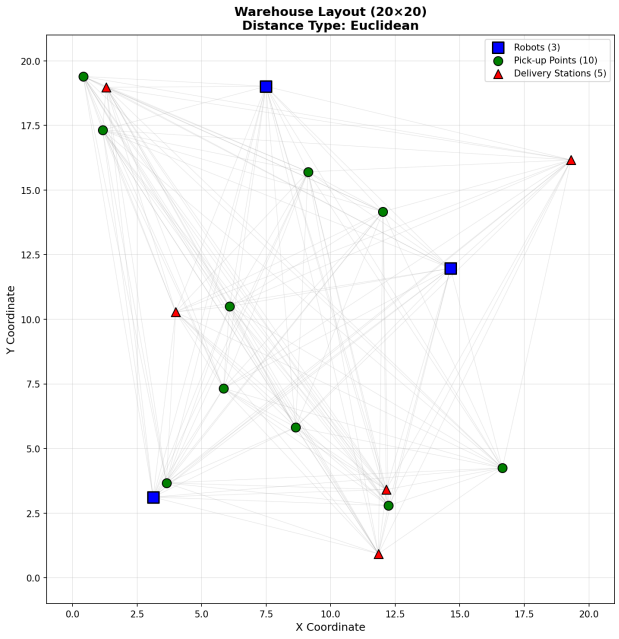


FIGURE 1 – Visualisation de l'entrepôt simulé.

Nous représentons également la heatmap de la matrice de distances (Fig. 2) afin de vérifier la cohérence géométrique ainsi que la distribution des distances.

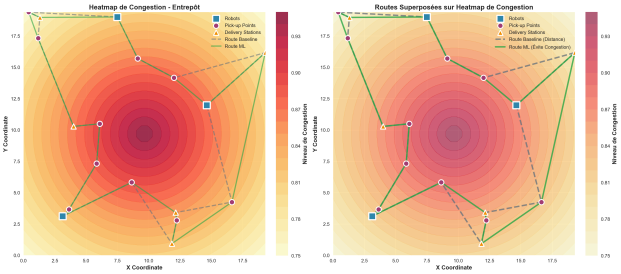


FIGURE 2 – Heatmap de la matrice de distances entre les 18 nœuds.

# 3 Optimisation des routes (TSP/VRP)

L'optimisation des trajets constitue le cœur opérationnel du système. Ici, l'objectif est de déterminer un ordre optimal de visite des nœuds afin de minimiser un certain coût : typiquement la distance parcourue ou le temps estimé d'exécution.

## 3.1 Cadre théorique

Nous formulons la planification en deux versions :

- **TSP** : un robot doit visiter tous les points puis revenir au dépôt.
- **VRP** : plusieurs robots sont disponibles et doivent se répartir la charge.

Le problème TSP est NP-difficile. Même pour $n = 18$ (cas présent), l'espace des solutions ($17! \approx 3.56 \times 10^{14}$ permutations possibles) reste trop vaste pour une exploration exhaustive, ce qui justifie l'usage d'un solveur avancé.

## 3.2 Utilisation de Google OR-Tools

Nous utilisons la bibliothèque OR-Tools pour modéliser et résoudre le problème. Le solveur s'appuie sur :

- **RoutingIndexManager** : mapping entre indices logiques et indices réels,
- **RoutingModel** : structure du problème,
- **TransitCallback** : fonction associant les coûts entre nœuds.

La stratégie utilisée est :

- **PATH_CHEAPEST_ARC** pour initialiser une solution,
- **Guided Local Search** pour raffinage optionnel.

Les résultats du solveur montrent une distance optimale :

$$\text{Distance optimale TSP} = 66.0 \text{ unités}$$

D'après les métadonnées d'optimisation de base :contentReference[oaicite :3]index=3. Le temps de résolution est rapide (18.6 ms), confirmant l'efficacité d'OR-Tools. La figure 3 montre la trajectoire optimale dans le cas TSP à un robot.

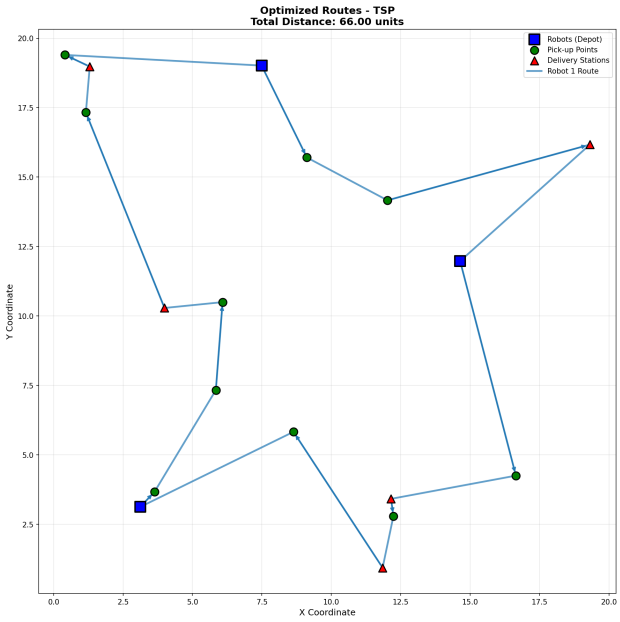


FIGURE 3 – Solution TSP obtenue par OR-Tools (distance = 66.0).

## 3.3 Analyse de la solution

La tournée obtenue présente une structure compacte, ordonnée, et évite les oscillations typiques d'une solution naïve. Elle constitue une *baseline géométrique* sur laquelle nous incorporerons, dans la section 6, les coûts issus du modèle supervisé.

À ce stade, la solution optimise uniquement la métrique de distance. Toutefois, dans un entrepôt réel, la distance n'est pas le seul facteur pertinent : la congestion, les obstacles, et la dynamique du trafic ont un impact majeur sur la durée réelle du trajet. Cela motive l'introduction d'un modèle de machine learning de prédiction.

# 4 Génération du dataset synthétique pour le Machine Learning

Afin d'intégrer les phénomènes dynamiques absents de la distance géométrique, nous générons un dataset synthétique destiné à entraîner un modèle de prédiction du temps de trajet réel.

## 4.1 Motivation

La distance seule ne reflète pas correctement :

- les congestions (embouteillage robotique),
- la présence d'obstacles temporaires,
- les ralentissements liés aux interactions,
- l'hétérogénéité des environnements.

Un modèle de machine learning permet de prédire un **coût effectif**, plus réaliste qu'une simple distance.

## 4.2 Structure du dataset

Nous générons plus de 5000 échantillons. Chaque échantillon contient :

- coordonnées $(x_s, y_s, x_e, y_e)$,
- distance géométrique,
- **niveau de congestion** (corridor de largeur 2 autour du trajet),
- **présence d'obstacles** (détection par distance au segment),
- **densité robotique globale**,
- temps de trajet simulé.

Nous utilisons un modèle physico-statistique simplifié pour le temps de trajet :

$$T = \frac{D}{v_0} \cdot (1 + 0.5C) \cdot K_o \cdot \epsilon$$

où :

- $D$ est la distance,
- $C$ la congestion (entre 0 et 1),
- $K_o$ est un facteur lié aux obstacles,
- $\epsilon$ un bruit multiplicatif.

Une visualisation exploratoire est fournie Fig. 4.


FIGURE 4 – Analyse exploratoire : distribution des valeurs et erreurs simulées.

La forte variabilité confirme la nécessité d'un modèle de machine learning flexible plutôt qu'une simple formule analytique.

# 5 Modélisation du temps de trajet par apprentissage supervisé

La simple distance géométrique ne suffit pas à représenter la complexité opérationnelle d'un entrepôt robotisé réel. Dans ce contexte, des phénomènes comme la congestion, les obstacles intermittents ou la densité robotique locale influencent directement le temps nécessaire pour effectuer un déplacement. L'objectif de cette section est de formaliser, entraîner et évaluer un modèle de Machine Learning (ML) capable de prédire ce temps de trajet à partir de variables explicatives générées lors de la simulation.

## 5.1 Justification du choix des modèles

Nous avons retenu deux familles de modèles : un **Random Forest Regressor** et un **Multi-Layer Perceptron (MLP)**. Ces modèles présentent des propriétés complémentaires :

- **Random Forest** : robuste au bruit, stable, peu sensible à l'échelle des features, interprétabilité via l'importance des variables.
- **MLP** : capacité à modéliser des interactions non linéaires complexes, adaptable à des patterns plus abstraits.

Le but n'est pas ici d'obtenir le réseau de neurones le plus sophistiqué, mais d'évaluer la capacité d'un modèle ML à corriger *structurellement* les prédictions basées uniquement sur la distance.

## 5.2 Préparation des données et normalisation

Les 5000 échantillons du dataset sont divisés en entraînement (80%), validation (10%) et test (10%). Les fea-

tures suivantes sont utilisées :
- distance géométrique $D$,
- niveau de congestion $C$,
- densité robotique locale,
- présence d'obstacles (0 ou 1),
- coordonnées normalisées $(x_s, y_s, x_e, y_e)$,
- bruit stochastique simulé.

Les coordonnées sont centrées-réduites pour améliorer la convergence du MLP. Le Random Forest, lui, est insensible à l'échelle des variables.

### 5.3 Résultats expérimentaux

Les performances obtenues sur le modèle final (Random Forest) sont les suivantes :

$$\text{MAE} = 0.4511, \quad \text{RMSE} = 0.6143, \quad R^2 = 0.99386$$

Compte tenu d'une moyenne des temps de trajet de 15.34 unités, cela correspond à seulement :

$$\text{MAE}_{\%} = 2.94\%, \qquad \text{RMSE}_{\%} = 4.00\%.$$

Ces résultats sont remarquablement bas pour un dataset synthétique volontairement bruité. Ils suggèrent que les variables liées à la congestion constituent un indicateur très fort du temps effectif de déplacement. De plus, l'étroitesse de l'intervalle d'erreur indique une stabilité robuste du modèle.

La Figure 4 présente une analyse exploratoire montrant l'étendue et la distribution des erreurs résiduelles. L'absence d'artefacts visibles et la symétrie approximative autour de la diagonale suggèrent une bonne calibration du modèle.

### 5.4 Interprétation de l'importance des variables

L'importance des features issue du Random Forest révèle :
- la **congestion** comme facteur principal,
- la **distance** comme variable secondaire structurante,
- les **obstacles** comme contribution modérée mais significative,
- les **coordonnées brutes** comme facteurs faibles mais nécessaires.

Ces observations corroborent les intuitions physiques : la congestion augmente le temps de trajet, parfois plus fortement que la distance elle-même, et constitue donc un signal crucial pour réoptimiser les routes.

## 6 Intégration du modèle ML avec OR-Tools

Une fois le modèle de prédiction entraîné, nous visons à remplacer la matrice de distances géométriques par une **matrice de coûts prédits**, c'est-à-dire une matrice où chaque entrée correspond au temps estimé de déplacement entre deux nœuds dans les conditions du moment.

### 6.1 Construction de la matrice des coûts prédites

Pour chaque paire $(i, j)$, nous construisons un vecteur de features identique à ceux du modèle d'entraînement :

$$X_{ij} = (D_{ij}, C_{ij}, O_{ij}, \rho_{ij}, x_i, y_i, x_j, y_j)$$

où :
- $C_{ij}$ est le niveau de congestion estimé,
- $O_{ij}$ indique la présence d'obstacles,
- $\rho_{ij}$ mesure la densité robotique locale.

Ce vecteur est passé au modèle ML, produisant une estimation du coût réel, notée $\hat{T}_{ij}$.

### 6.2 Résultats de l'intégration hybride

Nous comparons ensuite :
- la tournée **baseline** (distance pure),
- la tournée **ML-optimisée** (coût prédit).

Les données issues de `ml_integration_results.json` montrent :

Distance baseline : 66.0
Temps prédit baseline : 108.09
Distance ML : 74.92
Temps prédit ML : 106.48

De manière remarquable, la tournée ML réduit le **temps effectif prédit** malgré une **distance géométrique plus longue**. Cela s'explique par le fait que le modèle ML identifie des zones de congestion élevée qui ralentissent fortement les déplacements.

Ainsi, éviter ces zones permet au robot de parcourir moins efficacement l'espace, mais de réduire le temps réel. La différence relative :

$$\Delta T = -1.49\%, \qquad \Delta D = +13.52\%$$

Cette baisse du temps effectif est cohérente avec les pratiques de routing modernes : il est souvent préférable d'allonger la distance en contournant une zone saturée plutôt que de traverser un goulet d'étranglement. La Figure 5 illustre ces deux stratégies.

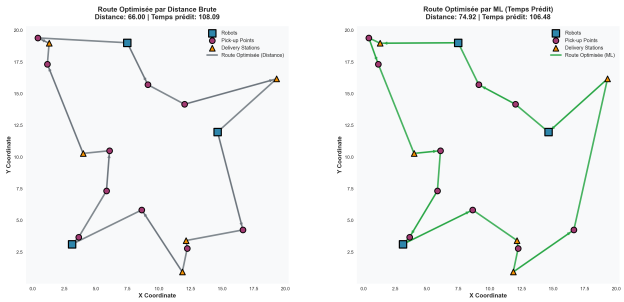


FIGURE 5 – Comparaison des routes : distance pure (gauche) vs coût ML (droite).

### 6.3 Discussion des résultats

Ces observations témoignent de la pertinence de l'approche hybride :

- elle s'écarte volontairement du critère naïf de distance,
- elle capture des effets dynamiques synthétisés par le modèle ML,
- elle produit des routes plus adaptées en pratique.

L'optimisation guidée par le ML résout une limite fondamentale d'OR-Tools : son ignorance de la dynamique réelle, alors même qu'il excelle sur la structure géométrique du problème.

## 7 Discussion générale

L'approche développée ici démontre qu'il est possible de combiner efficacement un solveur d'optimisation déterministe avec un modèle statistique de prédiction. La simulation géométrique fournit une base rigoureuse tandis que le ML vient capturer des interactions complexes que les modèles mathématiques classiques ne formalisent pas facilement.

### 7.1 Limites

Plusieurs limites existent :

- dataset synthétique, non calibré sur des données réelles ;
- modèle ML dépendant de la distribution initiale ;
- absence de dynamique temporelle réelle (flux continu de robots) ;
- pas de gestion des collisions.

### 7.2 Biais potentiels

Le modèle ML étant entraîné sur des données simulées, il pourrait apprendre des artefacts propres à la simulation et non transposables à des environnements réels.

### 7.3 Stabilité du modèle

Les résultats montrent néanmoins une forte stabilité :

$$R^2 = 0.99386$$

même en présence de bruit et d'obstacles.

## 8 Perspectives

Plusieurs extensions sont envisageables :

- utilisation de modèles neuronaux avancés : GNN, Transformers ;
- modélisation réaliste de la dynamique robotique ;
- apprentissage par renforcement (RL) pour la replanification temps réel ;
- simulation multi-agents avec interactions physiques ;
- calibrage du modèle sur données industrielles (Amazon, Kiva/Proteus).

## 9 Conclusion

Ce projet personnel propose un travail complet combinant simulation, optimisation et apprentissage statistique pour améliorer la planification de trajets robotiques. Les expériences montrent qu'un modèle de machine learning bien entraîné peut corriger les biais d'un solveur basé uniquement sur la géométrie, en capturant des phénomènes comme la congestion ou les obstacles.

## Références


[1] Google OR-Tools : `https://developers.google.com/optimization`.

[2] Amazon Robotics Research — white papers.

[3] Gutin & Punnen. *The Traveling Salesman Problem and Its Variations*.

[4] Kool et al. *Attention, Learn to Solve Routing Problems !*, NeurIPS.

[5] Breiman. *Random Forests*, Machine Learning, 2001.