Présentation

OptiPick — Allocation intelligente de commandes en entrepôt

Notre objectif est de résoudre un problème d'optimisation logistique : comment affecter des commandes à différents agents en respectant des contraintes métier, tout en optimisant un critère global.

Problème

Comment assigner des commandes à des agents en respectant :

Capacité poids / volume

Zones interdites

Produits fragiles

Poids max par item

Incompatibilités produits

Objectifs :

- Maximiser le nombre de commandes assignées

OU

- Minimiser le coût total

- C’est un problème d’allocation sous contraintes, typique des CSP (Constraint Satisfaction Problems)

Architecture du système

Étapes

- Chargement JSON
- Parsing en objets Python
- Enrichissement (poids total, volume, localisation)
- Allocation (glouton ou MiniZinc)
- Affichage des résultats

- Le projet est structuré proprement : séparation données, modèle, contraintes et solveur.
Cela permet de changer d’algorithme sans modifier toute l’architecture.

Modélisation des données

Modèle conceptuel

- Agent
- Order
- Product
- Warehouse

- assignment[order_i] = agent_j : Chaque commande choisit un agent.

- On modélise le problème sous forme déclarative : on ne dit pas comment affecter, on décrit les contraintes que doit respecter la solution.

Algorithme glouton : First-Fit

Principe

- On parcourt les commandes
- On assigne au premier agent disponible

Résultat

- 30/30 commandes assignées
- Distance proxy = 594

- Avantages : Rapide - Simple
- Limites : Ne vérifie pas toutes les contraintes -Pas optimal

Algorithme glouton : First-Fit

```
JOUR 1 — Allocation naïve (First-Fit)
═══════════════════════════════════════

Commandes totales : 30
Commandes assignées: 30
Commandes non assignées: 0
Distance totale estimée (proxy): 594

Détail par agent:
- R1 (robot)
  commandes: 7 -> ['Order_001', 'Order_002', 'Order_003', 'Order_004', 'Order_005', 'Order_006'
 'Order_022']
  poids: 18.64/20.00 kg (93.2%)
  volume: 29.74/30.00 dm³ (99.1%)
  vitesse: 2.0 m/s

- R2 (robot)
  commandes: 5 -> ['Order_007', 'Order_008', 'Order_009', 'Order_010', 'Order_023']
```

MiniZinc & CSP
Constraint Satisfaction Problem

CSP

Variable - Domaines - Contraintes

MiniZinc

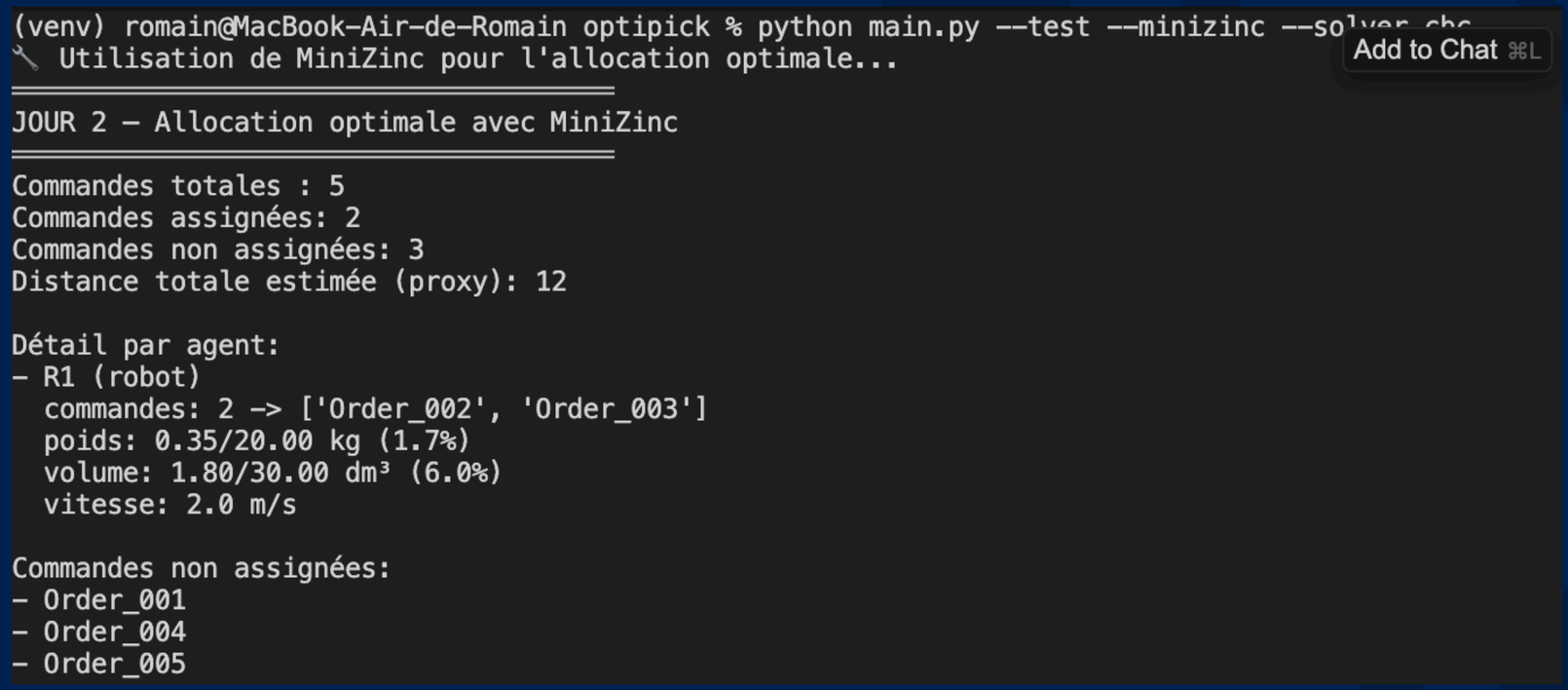
- Modélisation déclarative
- Solveur explore l’espace des solutions

Résultat

- 9/10 commandes assignées
- Toutes contraintes respectées
- Meilleure validité métier

- Contrairement au glouton, MiniZinc explore les combinaisons possibles et garantit que toutes les contraintes sont respectées.

Modélisation des contraintes MiniZinc



Explication des contraintes

```
% 3. Restrictions des robots (zones interdites)
% Note: order_zones[order_idx] = 0 peut signifier Zone A ou zone non définie (par défaut Zone A)
% On vérifie seulement si la zone est dans la plage valide [0..4]
constraint forall(order_idx in ORDERS, agent_idx in AGENTS) (
    (assignment[order_idx] == agent_idx /\
    agent_type[agent_idx] == 0 /\
    order_zones[order_idx] >= 0 /\
    order_zones[order_idx] < n_zones) ->
    (not forbidden_zones[agent_idx, order_zones[order_idx]])
);
```

Explication des contraintes

```
% 4. Restrictions des robots (pas d'objets fragiles)
constraint forall(order_idx in ORDERS, agent_idx in AGENTS) (
    (assignment[order_idx] == agent_idx /\ agent_type[agent_idx] == 0 /\ no_fragile[agent_idx]) ->
    (not order_has_fragile[order_idx])
);
```

Explication des contraintes

```
% 5. Restrictions des robots (poids max par item)
constraint forall(order_idx in ORDERS, agent_idx in AGENTS) (
    (assignment[order_idx] == agent_idx /\ agent_type[agent_idx] == 0 /\ max_item_weight[agent_idx] > 0) ->
    (order_max_item_weight[order_idx] <= max_item_weight[agent_idx])
);
```

Explication des contraintes

```
% 6. Incompatibilités entre produits
% Si deux commandes sont incompatibles, elles ne peuvent pas être assignées au même agent
constraint forall(order_first in ORDERS, order_second in ORDERS where order_first < order_second /\
    incompatible[order_first, order_second]) (
    assignment[order_first] != assignment[order_second] \/
    assignment[order_first] == 0 \/
    assignment[order_second] == 0
);
```

Merci à tous !