HEC LIÈGE
Management School - Liège Université

Le Deep Reinforcement Learning à la rescousse des problèmes combinatoires

Une nouvelle approche pour le 3DBPP en logistique

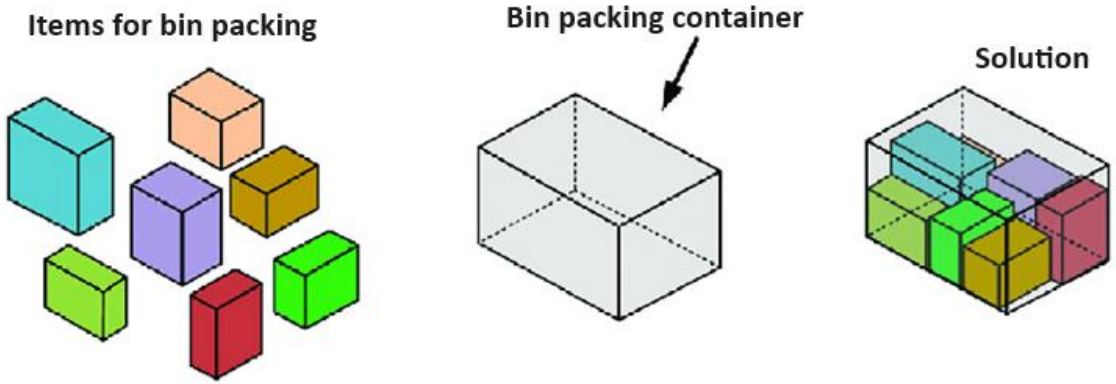
Groupe ADP – Novembre 2023
Evers Justine & Michaël Schyns

Table des matières

- Introduction
- Reinforcement Learning
- Reinforcement Learning pour le 3DBPP
- Architecture du réseau de neurones
- Apprentissage
- Multi-bin
- Expériences et résultats
- Conclusion and futures directions

Introduction

- 3D Bin Packing Problem
  - Ensemble de conteneurs faiblement hétérogène
  - Ensemble de boites cuboïdes fortement hétérogène
  - Placer toutes les boites dans un nombre minimum de conteneurs
  - Premières contraintes : non-chevauchement des différents objets et de stabilité

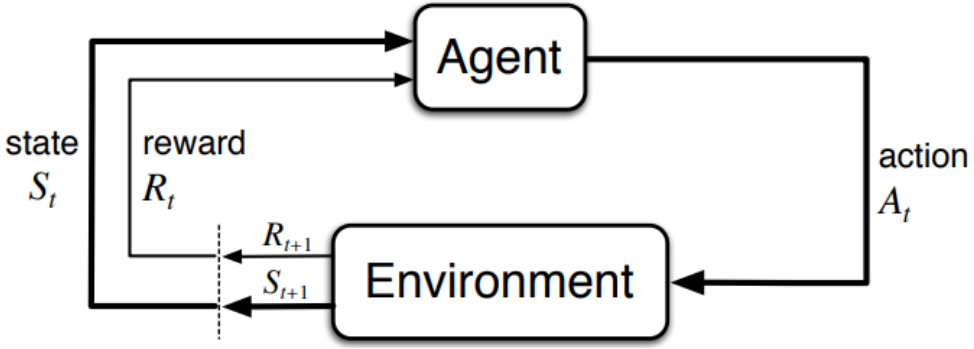


Introduction

- Méthodes exactes ?
  - ➔ Ne conviennent pas aux grandes instances
- Heuristiques ?
  - ➔ Disponibles pour le 2DBPP
  - ➔ La troisième dimension ajoute un niveau de complexité important
- Intelligence Artificielle et Machine Learning
  - ➔ Mais pas initialement développées pour des problèmes combinatoires

Introduction

- Reinforcement Learning (RL)
- **L'agent** effectue des **actions**, ce qui entraine un nouvel **état** de **l'environnement** et déclenche une **récompense**



- L'agent apprend (automatiquement) à reconnaitre et à prendre de bonnes actions au fil du temps

Introduction

- **Challenge** : Le RL n’est pas conçu pour des problèmes combinatoires avec d’immenses espaces d’états

- **Solution** : Le Deep Learning (DL) peut être utilisé pour construire des modèles capables de représenter des états

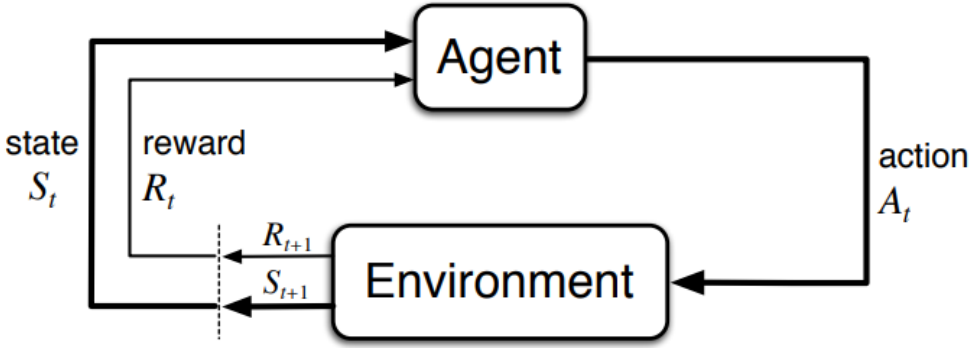
- **Conclusion** : Le Deep Reinforcement Learning (DRL) est une approche prometteuse

Introduction

- Revue de la littérature: le DRL n'apparait que dans quelques (récents) articles portant sur le 3DBPP
- Parmi eux, celui rédigé par **Que et al. (2023)** semble très prometteur
  - Limitations : 1 seul conteneur avec une hauteur infinie et des contraintes basiques (i.e., non-chevauchement des différents objets)
  - C'est le cas de la plupart de la littérature sur le sujet (Ali et al., 2022)
- Notre contribution
  - Multi-bin
  - Stabilité verticale et choix d'un ensemble pertinent de contraintes
  - Solution complète de la théorie à la pratique: outil en RA pour implémenter une solution

Introduction

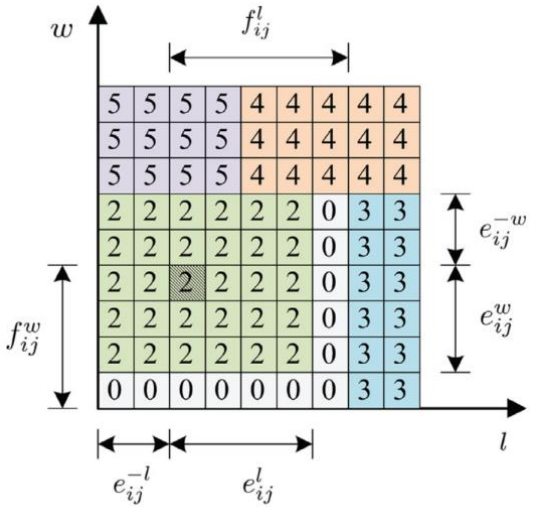
- Reinforcement Learning (RL)
- **L'agent** effectue des **actions**, ce qui entraine un nouvel **état** de **l'environnement** et déclenche une **récompense**



- L'agent apprend (automatiquement) à reconnaitre et à prendre de bonnes actions au fil du temps

RL pour le 3DBPP – Espace des états

- s_t est composé de 2 canaux
  - ➔ Le **box state** défini comme $s_b = \{b_1, b_2, \dots, b_n\}$, où $b_i = (l_i, w_i, h_i)$ réfère à la dimension de la boite i, et n est le nombre de boites qui n'ont pas encore été chargées
  - ➔ Le **container state** défini comme une matrice s_c aux dimensions $W \times L$, dont les éléments sont les vecteurs $c_{ij} = \left(h_{ij}, e_{ij}^{l}, e_{ij}^{w}, e_{ij}^{-l}, e_{ij}^{-w}, f_{ij}^{l}, f_{ij}^{w}\right)$ qui contiennent de l'information à propos de la position p_{ij}



RL pour le 3DBPP – Espace des actions

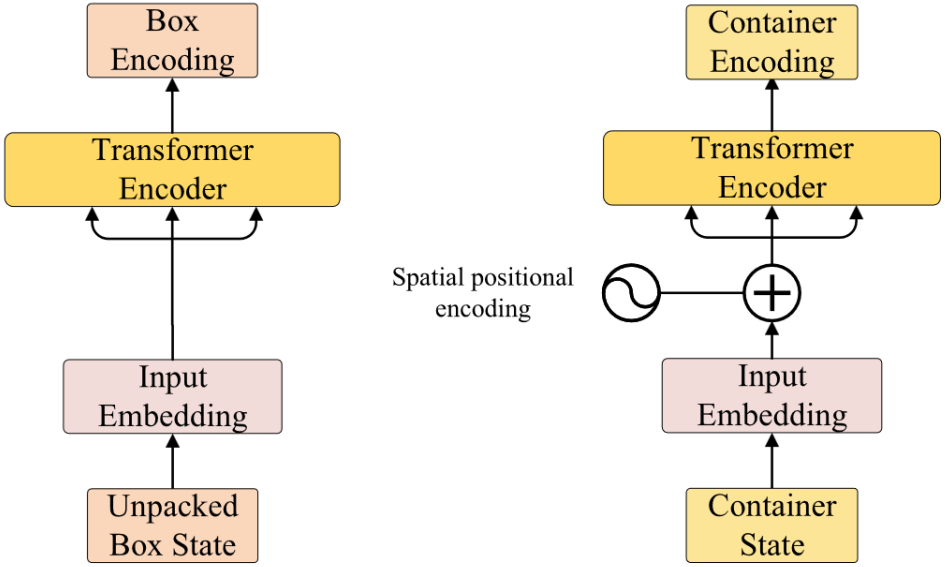
- 3 décisions à prendre
  1. Position
  2. Sélection de la prochaine boite à charger
  3. Orientation
- $a_t = \left(a_t^p, a_t^s, a_t^o\right)$
- Définition de la policy
  - $\pi(a_t|s_t) = \pi\left(a_t^p, a_t^s, a_t^o|s_t\right) = \pi\left(a_t^p|s_t\right)\pi\left(a_t^s|a_t^p, s_t\right)\pi\left(a_t^o|a_t^p, a_t^s, s_t\right)$
  - Chain Rule
  - Espace des actions réduit
- Les actions interdites sont rendues impossibles via un **masking** de l'espace des actions dans la partie deep learning (voir plus loin)

RL pour le 3DBPP – Fonction de récompense

- $r_t = gap_{t-1} - gap_t$
  - ➔ $gap_t = WL\widetilde{H_t} - \sum_i w_i l_i h_i$ où $\widetilde{H_t}$ fait référence à la hauteur des boites chargées au temps t, et $\sum_i w_i l_i h_i$ fait référence au volume total des boites déjà chargées dans le conteneur au temps t

Architecture du réseau de neurones

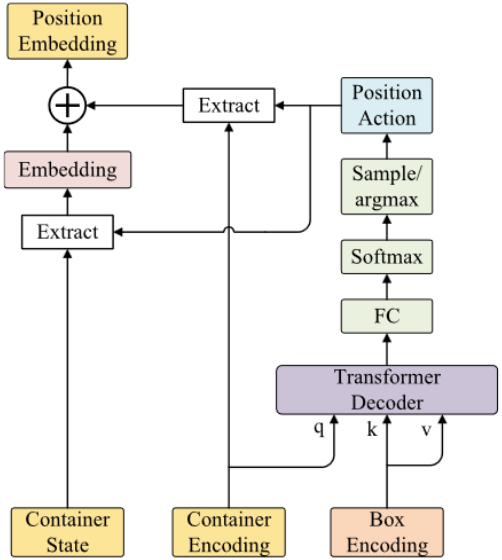
- Transformers avec encoders et decoders



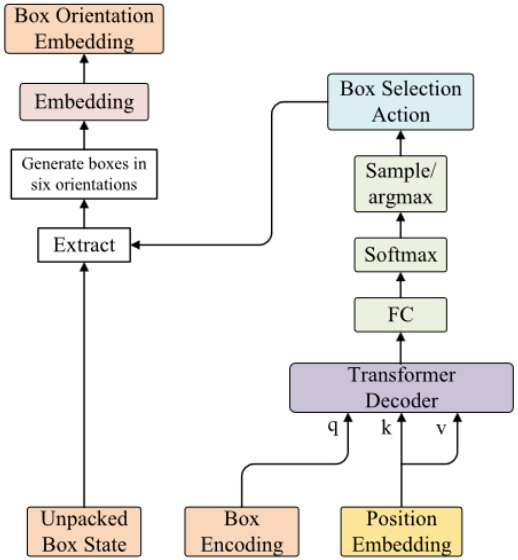
(a) Box encoder (b) Container encoder

Architecture du réseau de neurones

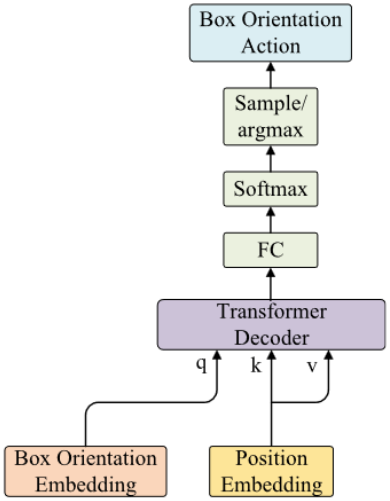
- Transformers avec encoders et decoders



(a) Position decoder



(b) Selection decoder



(c) Orientation decoder

Apprentissage

- Les contraintes sont imposées par l'intermédiaire d'un **masque**
  - Représentation de toutes les actions possibles
  - Tableau à 4 dimensions
    - Pour chaque boîte (dimension 1)
    - Chacune des 6 orientations possibles (dimension 2)
    - Une matrice booléenne de dimension $W \times L$ correspondant à chaque position (x, y) possible du conteneur (dimension 3 et 4)
  - Dans l'approche de Que et al. (2023), le masque ne gère que les limites du conteneur

Apprentissage

- Appelons (b, r, x, y) un élément particulier de ce tableau
- Si…
  - Le coin inférieur arrière gauche de la **boite b**
  - Avec la **rotation r** peut, par rapport aux dimensions du conteneur
  - Être placé aux **coordonnées** $(\boldsymbol{x}, \boldsymbol{y})$

➔ $(b, r, x, y) = True$

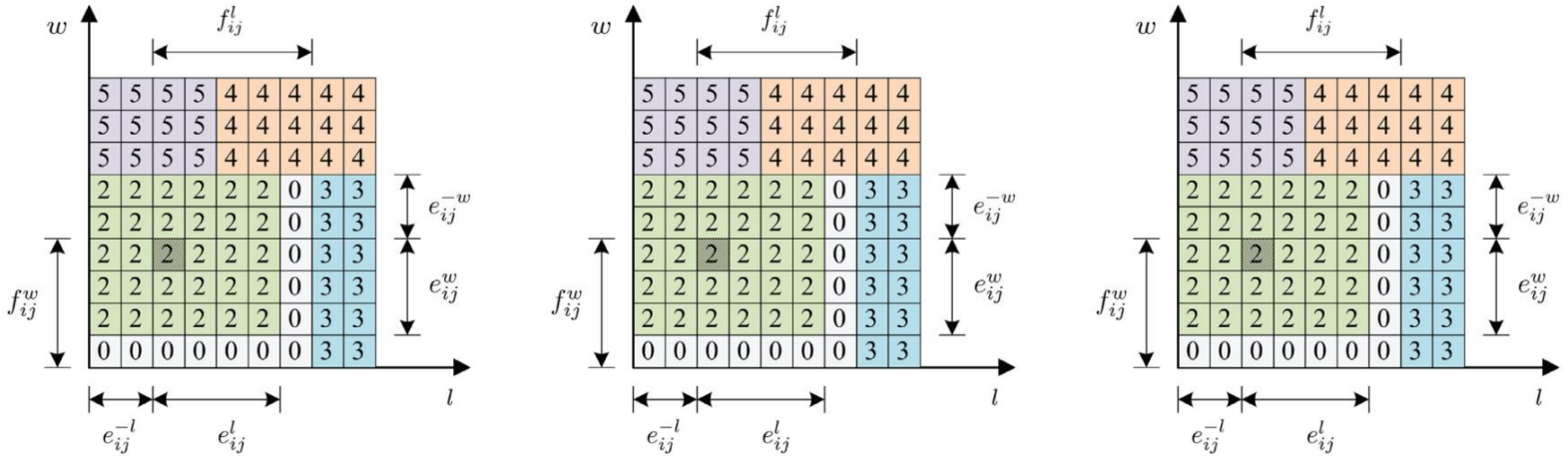
Apprentissage

- Initialement le masque ne fait pas partie des entrées des 3 decoders
- Les probabilités de sortie ne tiennent pas compte du masque
- Elles sont modifiées ultérieurement

  ➔ **<u>Toute action incompatible aura une probabilité de 0 !</u>**
- Si toutes les probabilités = 0 ➔ l'épisode s'arrête
- Le non-chevauchement n'est pas garanti par le masque !
  - Les autres boites ne sont pas prises en compte
  - MAIS la contrainte est implicitement respectée grâce à la stratégie de mise-à-jour de la matrice des états et de reward

Multi-bin

- Plusieurs matrices s_c



- Limite sur la valeur de h_{ij}
- Plus tard : intégration de la stabilité verticale, et autres contraintes

Expériences

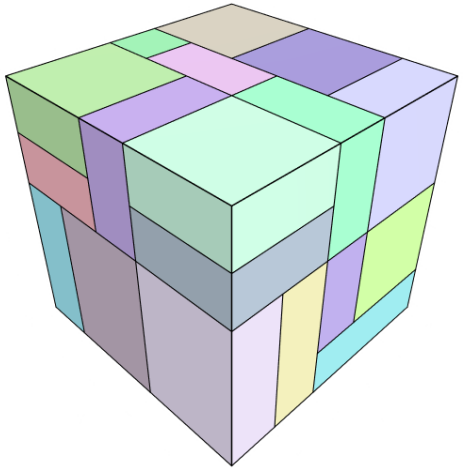
- Notre dataset initial : guillotine-cuts (de différentes tailles) dans un cube d'une dimension donnée
- Par construction, il existe au moins une solution parfaite (sans vide)
  Possibilité de mesurer la qualité de l'approche
- Exemple : cubes $5 \times 5 \times 5$ et 96 boites

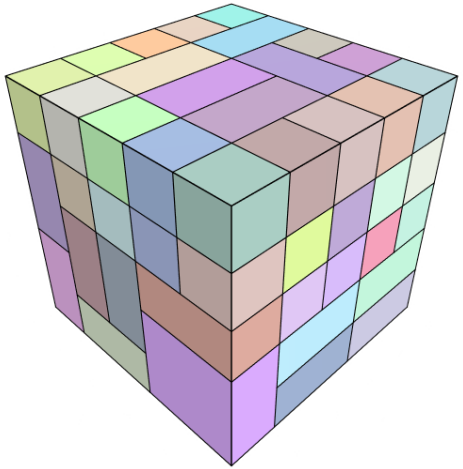
Résultats

- Résultat (best case) :

Résultats

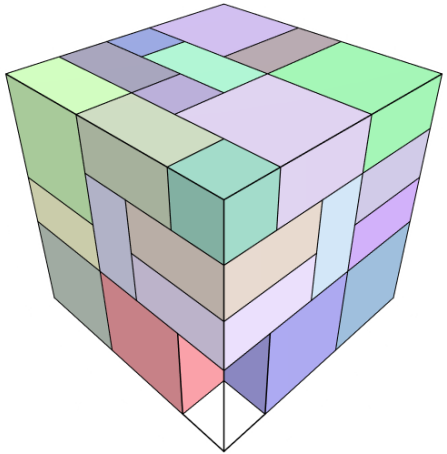
- Dataset de 1024 instances générées aléatoirement
- Cubes $5 \times 5 \times 5$ et 96 boites
  - Meilleur taux de remplissage : 100%

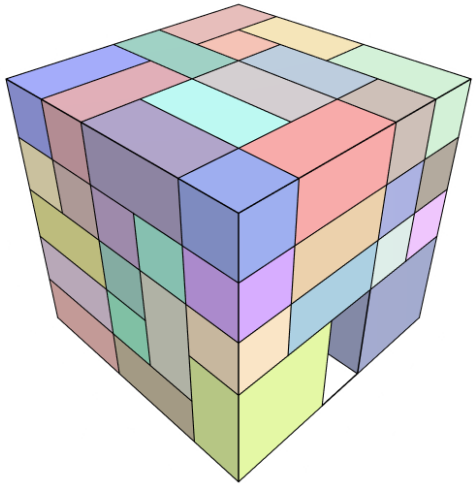


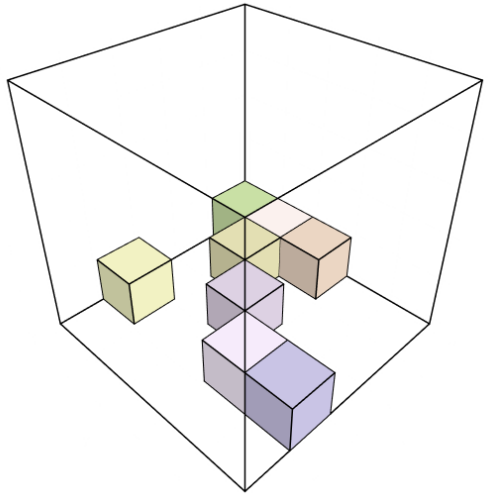


Résultats

- Dataset de 1024 instances générées aléatoirement
- Cubes $5 \times 5 \times 5$ et 96 boites
  - Pire taux de remplissage : 66,67%







Résultats

- Dataset de 1024 instances générées aléatoirement
- Cubes $5 \times 5 \times 5$ et 96 boites
  - Meilleur taux de remplissage : 100%
  - Pire taux de remplissage : 66,67%
  - Taux de remplissage moyen : 83,05%

Conclusions et directions futures

- Cette recherche se concentre sur l'utilisation d'une technique issue de la Data Science pour résoudre le 3DBPP
- Le Reinforcement Learning est une technique dans laquelle un agent apprend automatiquement à reconnaître et à prendre de bonnes actions au fil du temps
- Compte tenu de la nature combinatoire du 3DBPP, le RL seul ne peut être utilisé, et le Deep Reinforcement Learning vient à la rescousse
- Notre revue de la littérature montre que les articles publiés sur le sujet jusqu'à présent ont encore des limites
- Nous avons développé un nouveau modèle prenant en compte le multi-bin
- Nous prévoyons d'incorporer plus de contraintes dans notre modèle
- Outil en Réalité Augmentée

Contact

- Justine Evers
- Doctorante à HEC-Liège
- jevers@uliege.be

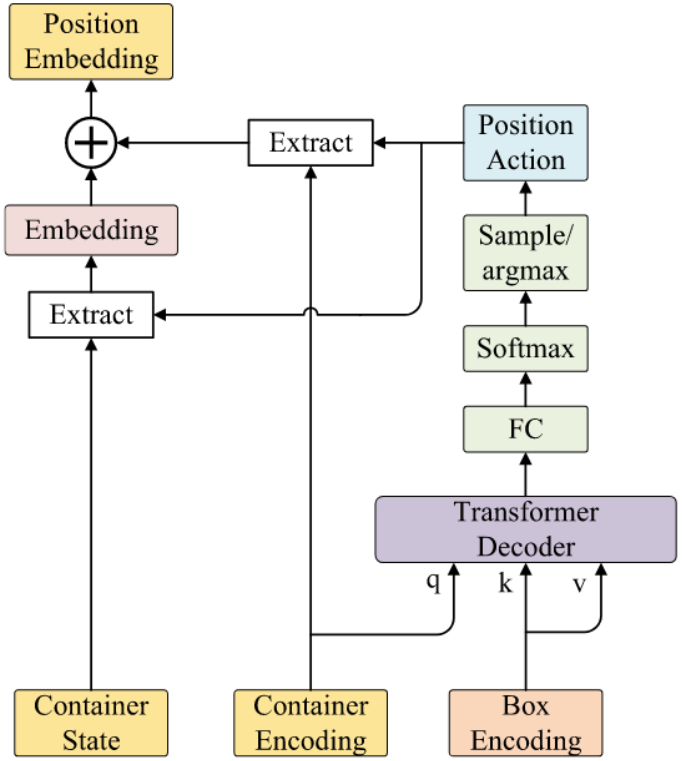
Bibliographie

- Ali, S., Ramos, A. G., Carravilla, M. A., & Oliveira, J. F. (2022). On-line three-dimensional packing problems: A review of off-line and on-line solution approaches. *Computers & Industrial Engineering*, *168*, 108122.

- Que, Q., Yang, F., & Zhang, D. (2023). Solving 3D packing problem using Transformer network and reinforcement learning. *Expert Systems with Applications*, *214*, 119153.

Position Decoder

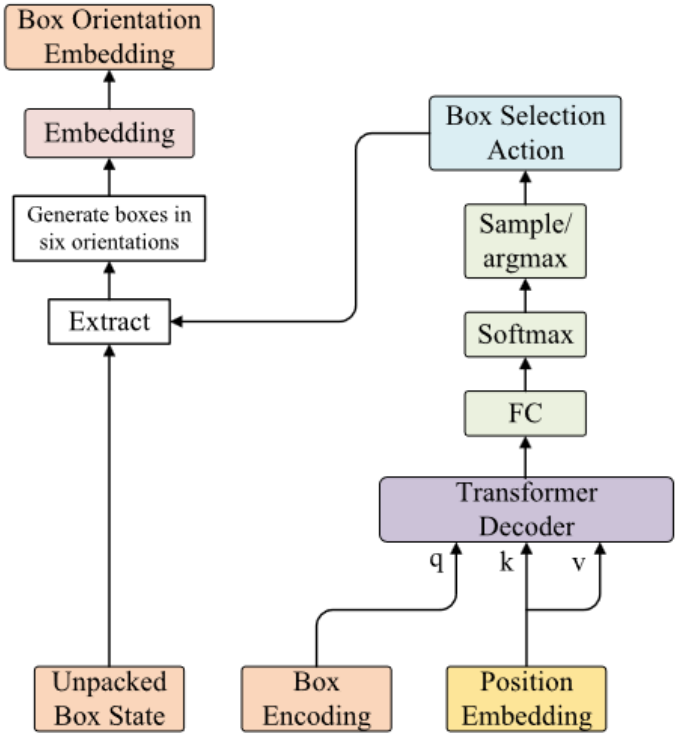
- Entrée
  - Box state
  - Container state
  - ⇔ Sortie des 2 encoders
- Sortie
  - Distribution de probabilité sur chacune des positions de la projection du conteneur sur le plan (x, y)
  - Indique la désirabilité de chacune de ces positions



(a) Position decoder

Selection Decoder

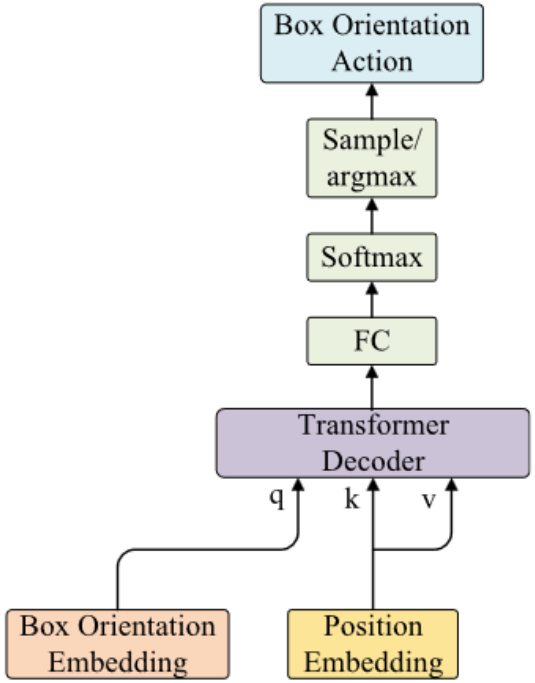
- Entrée
  - Box state
  - La position choisie
- Sortie
  - Distribution de probabilité sur la liste initiale de boites
  - Indique la désirabilité de chacune de ces boites, sachant la position qui a été choisie



(b) Selection decoder

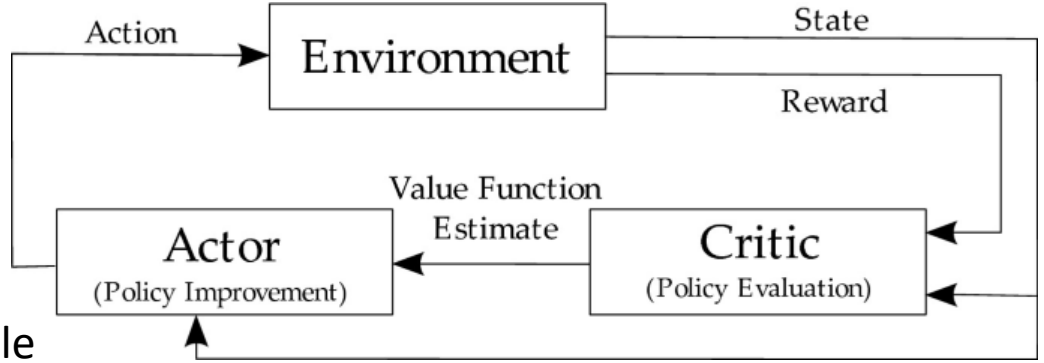
Orientation Decoder

- Entrée
  - La boite choisie
  - La position choisie
- Sortie
  - Distribution de probabilité sur chacune des 6 orientations de la boite choisie
  - Indique la désirabilité de chacune de ces orientations, sachant la position choisie



(c) Orientation decoder

Actor-critic



- Actor
  - Explore l'environnement
  - Détermine la policy optimale
- Critic
  - Estime la valeur de décision prise par l'actor
  - Détermine si cette décision résultera en une meilleure récompense ou non
  - Guide l'actor

➔ L'actor reçoit le feedback du critic et ajuste sa policy pour améliorer ses perfromances