Intelligence Artificielle (I-INFO-026) -- Travaux Pratiques

Mini-projet N°01 : Application multi-agents pour la gestion des ventes de pièces automobiles

- **Introduction :**

L'objective de ce TP (ou mini-projet) est de d'augmenter l'intelligence d'un agent '**Courtier'** qui jouera le rôle d'intermédiaire entre des agents clients et vendeurs de pièces automobiles. L'agent « **Courtier** » devra chercher la meilleure offre parmi plusieurs offres proposées par trois agents du type « **Vendeur »** pour répondre à la demande de l'agent « **Acheteur »** avec le meilleur prix possible. Concrètement, le problème se présente comme suit :

- L'agent « **Acheteur »** envoie sa commande à un agent « **Courtier »** (agent intermédiaire), la commande se présente sous forme d'un objet qui contient le nom de la pièce auto demandée (exemple : plaquettes, suspensions, boîte de vitesse, essuie-glace, etc.) ainsi que le nombre de pièces désiré ;
- Chaque Vendeur est présenté par un agent indépendant (défini dans un script python séparé) dont chacun propose un service en fonction de la demande de l'agent « **Courtier »** et dispose d'un nombre défini de pièces ;
- Si le nombre demandé par l'agent « **Courtier** » (pour répondre au besoin de l'acheteur) est supérieur au nombre de pièces disponibles, la demande doit être refusée ;
- L'agent « **Vendeur** » doit répondre via un message du type **CFP** à l'agent « **Courtier »** en donnant une réponse indiquant la disponibilité ou pas de la pièce avec le prix. Si le nombre de pièces demandées dépasse 3, les agents « **Vendeur** » peuvent proposer une réduction (**25%** pour le **Vendeur1**, **20%** pour le **Vendeur2** et **25%** pour le **Vendeur3**).
- Afin de réaliser les différentes interactions, il faudra utiliser plusieurs actes de communication. Nous citons ici les plus importants :
    - **INFROM** : c'est le type de message le plus utilisé, il peut être employé dans plusieurs contextes comme (envoyer une information, confirmation ou un accusé de réception) ;
    - **REQUEST** : c'est un type de message qui permet de faire une demande initiale. L'agent qui envoie ce message doit recevoir une réponse de type INFORM par exemple ;

- **CFP** : c'est l'abréviation de (Call for Proposal), ce type de messages peut être utilisé lorsque les agents souhaitent demander une offre. Ce message doit recevoir comme réponse un message de type PROPOSE ;
- **PROPOSE** : comme son nom l'indique, c'est un message qui doit avoir comme contenu une proposition (soit un prix, un texte qui contient le nom d'un article, etc.). Ce message doit recevoir comme réponse soit « « **ACCEPT_PROPOSAL** » ou « **REFUSE** » ;
- **ACCEPT_ PROPOSAL** : pour accepter l'offre et terminer l'interaction ;
- **REFUSE** : pour refuser l'offre, l'agent qui reçoit ce type de message aura le droit de proposer une autre offre (PROPOSE). Ce type de message peut être remplacé par **REJECT_PROPOSAL** au cas où l'agent ne veut pas recevoir une nouvelle offre.

Exemples :

- Instruction pour la spécification d'un type de message ACL (pour un acte ***REQUEST***) :

  MyMessage = ACLMessage(ACLMessage.REQUEST)

- Instruction pour vérifier si le message reçu est de type **CFP** (Call For Proposal)

  if msg.performative == ''cfp'' *(toujours en minuscules)*

- **Enoncé du mini-projet :**

Télécharger le code de démarrage « **Starting_Code_TP2 »** depuis Moodle contenant la structure et le squelette du code à adapter/compléter pour répondre aux questions ci-dessous. La figure 1 illustre l'architecture des agents et interactions à réaliser.

- **Question 1 :** Compléter le code de **MainAgent.py**, pour démarrer les cinq Agents (Acheteur, Courtier, Vendeur_1, Vendeur_2 et Vendeur_3)
- **Question 2 :** Compléter le code des classes : **Courtier**, **Vendeur_1**, **Vendeur_2** et **Vendeur_3** de telle sorte que chacun puisse marquer son démarrage par un message d'accueil « **Bonjour je suis l'agent …** ». Compiler et tester le programme
- **Question 3 :** Au niveau de classe « **Acheteur** », encodez l'envoi suivant :

  L'agent **Acheteur** doit envoyer une demande à l'agent **Courtier** pour acheter une pièce de voiture, en initialisant statistiquement deux variables : nom de la pièce et le nombre d'unités souhaité. *(Le message doit avoir un format « objet » créé à l'aide la librairie pickle).* Le message envoyé doit avoir une ontologie ' *« **cmdAcheteur**' »* afin qu'elle soit reconnue par le **Courtier**.

- **Question 4 :** Au niveau de la classe « **Courtier** », encodez la réception et transfert du message comme suit :

  Une fois la demande de **l'Acheteur** reçue par le **Courtier**, ce dernier doit transférer l'objet de la demande aux trois agents vendeurs (créés précédemment). Il faudra développer les trois fonctions permettant de contacter les vendeurs une fois le message de l'acheteur avec une ontologie *« **cmdAcheteur** » reçu* (dans *react).*

  ➔ **Attention** :

  - Afin de bien observer l'interaction, l'envoi des messages doit être fait via le comportement « **call_latter** » afin d'appeler les 3 fonctions successivement avec un intervalle de temps de 3 secondes.
  - Il faudra aussi spécifier les ontologie « ***contactVend1*** », « ***contactVend2*** » et « ***contactVend3*** » aux messages envoyés par le Courtier aux trois vendeurs.
- **Question 5 :** au niveau des trois classes de vendeur « **Vendeur1, Vendeur2 et Vendeur3** » compléter le code pour leurs permettre de réagir aux demandes de courtier de type « **CFP** » et envoyer une réponse de type « **PROPOSE** » au « **Courtier** » avec le prix et avantage (promotion) proposés (les prix et avantages sont indiqués dans le code de démarrage).
- **Question 6 :** Au niveau de la classe « **Courtier** », encodez l'échange suivant :
  - Vérifier la bonne réception des messages de vendeurs.
  - Sélectionner l'offre selon la demande de l'Acheteur, puis calculer le prix total.
  - Envoyer un message de type ***ACCEPT_ PROPOSAL*** au bon vendeur et **REFUSE** au vendeur qui a proposé une mauvaise offre.
- **Question 7 :** Au niveau des 3 classes vendeur, encodez la confirmation de la vente (ou non) de la pièce en affichant un message (vente acceptée/refusée)
- **Question 8 :** Encoder la confirmation de la transaction par un envoi de message depuis le courtier vert l'acheteur et clôturer la demande.
- **Question 9 :** Améliorer le code de l'agent **Courtier** afin de prendre en considération le nombre des pièces demandés par l'Acheteur. Si ce nombre est supérieur ou égal à 3 pièces, le courtier peut appliquer une réduction sur le prix total en utilisant le taux envoyé par chaque vendeur. Développer le programme permettant de sélectionner la meilleure offre et afficher le prix final ainsi que le nom du vendeur

  Remarque : pour une meilleure simulation, donner la possibilité d'envoyer la même pièce par deux vendeurs différents mais à des prix/taux différents.

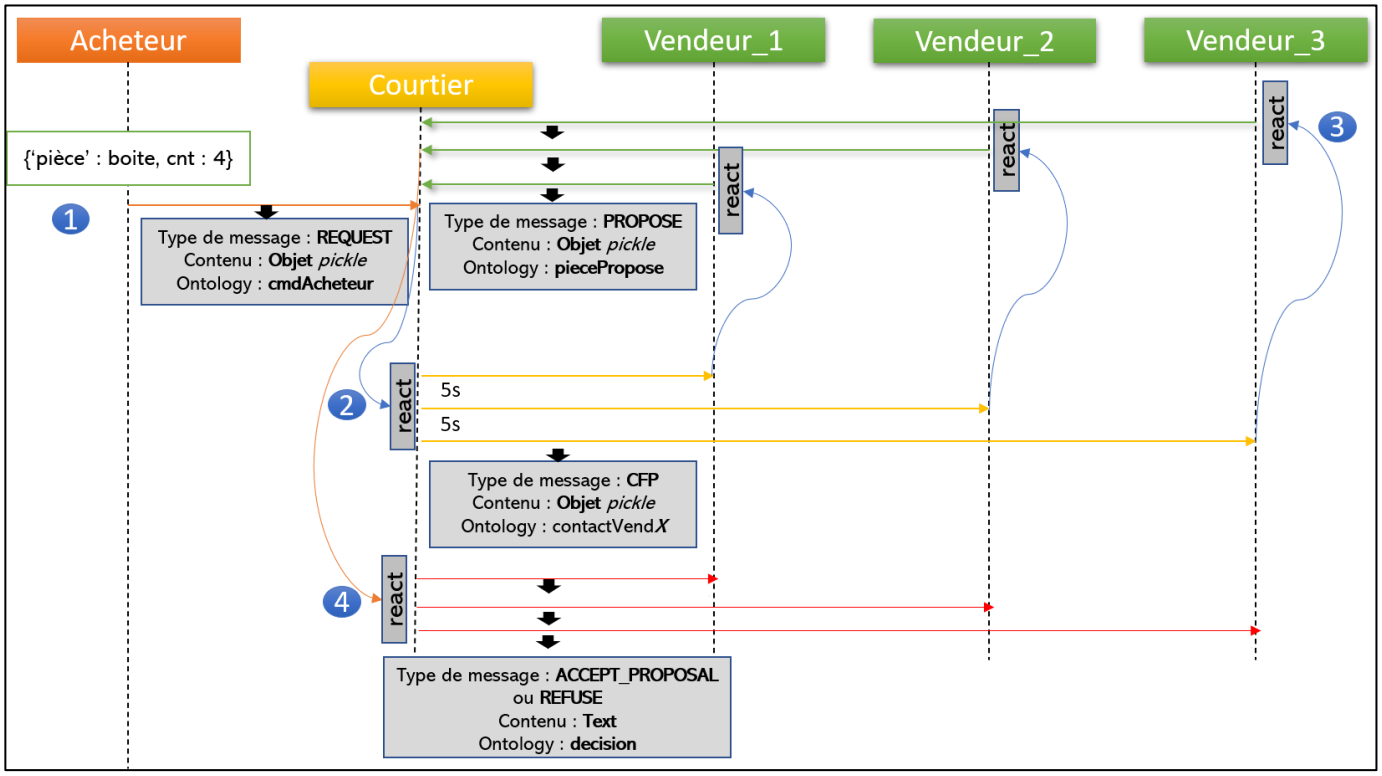

Figure 1 : Les interactions entre les différents agents

```
eladoui@el-adoui: ~/Bureau/pade/TP2/ExercicePiecesAuto

La commande recu de l'acheteur est :  plaquettes
La quantitie demande est :  3
Contact des vendeurs en cours ...

Contact vendeur N° 1 en cours ...
Vendeur_1 : Commande recu Tentative de vente de la piece boite-vitesse en cours ...
[courtier] 16/05/2020 12:37:28.617 --> Courtier !! : Message recu de : vendeur_1@localhost:2000
La piece proposee par le vendeur est :  boite-vitesse
Son prix à l'unité est : 38.98
Avantage en pourcentage pour 3 pieces ou plus :  25

!!!!!!!! Cette propostion ne correspond pas à la demande !!!!!!

Envoie de REJECT_PROPOSAL à  vendeur_1
Vendeur_1 : Tentative de vente refusée - peut être une autre fois

Contact vendeur N° 2 en cours ...
Vendeur_2 : Commande recu Tentative de vente de la piece plaquettes en cours ...
[courtier] 16/05/2020 12:37:31.610 --> Courtier !! : Message recu de : vendeur_2@localhost:2001
La piece proposee par le vendeur est :  plaquettes
Son prix à l'unité est : 29.98
Avantage en pourcentage pour 3 pieces ou plus :  20
Courtier : Reduction grace à l'avantage du vendeur est appliquée, nouveau prix est :  71.952  au lieu de :  89.94

****** Cette propostion correspond à la demande *****

Nombre de propositions similaires est  1
***** la meilleur offre est de  71.952  proposé par le vendeur :  vendeur_2


Contact vendeur N° 3 en cours ...
Vendeur_3 : Commande recu Tentative de vente de la piece plaquettes en cours ...
[courtier] 16/05/2020 12:37:36.613 --> Courtier !! : Message recu de : vendeur_3@localhost:2002
La piece proposee par le vendeur est :  plaquettes
Son prix à l'unité est : 33.98
Avantage en pourcentage pour 3 pieces ou plus :  25
Courtier : Reduction grace à l'avantage du vendeur est appliquée, nouveau prix est :  76.455  au lieu de :  101.94

****** Cette propostion correspond à la demande *****

Nombre de propositions similaires est  2
***** la meilleur offre est de  71.952  proposé par le vendeur :  vendeur_2
```

- **Questions facultatives :**

  - **Question 10 :** Améliorer le programme afin de prendre en considération le stock de pièces disponible chez chaque vendeur.

  - **Question11 :** Améliorer le programme de telle sorte qu'après confirmation de chaque commande, le courtier doit enregistrer le nom du vendeur avec le nom de la pièce. Si le client veut racheter la même pièce, le courtier doit réagir intelligemment et contacter directement le vendeur en question sans demander aux autres.

  - **Question 12 :** Améliorer encore plus le programme pour que l'agent Courtier puisse prédire automatiquement le nom du vendeur à contacter par rapport à la commande de l'acheteur en utilisant de nouveaux critères comme le prix, la qualité (marque), avis d'utilisateurs, etc.

  - **Question 13 :** afin de rendre le programme plus dynamique, proposez une interface graphique (tkinter, PyQt, etc.) permettant de récupérer les requêtes et réponses des agents à partir de des fichiers.

Réferences:

[1] González-Briones, Alfonso, et al. "Multi-agent systems applications in energy optimization problems: A state-of-the-art review." *Energies* 11.8 (2018): 1928.

[2] De Freitas, Bruna Kobay, et al. "Exploiting PADE to the Simulation of Multiagent Restoration Actions." *2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*. IEEE, 2019.

[3] Melo, Lucas Silveira, et al. "Python-based multi-agent platform for application on power grids." *International Transactions on Electrical Energy Systems* 29.6 (2019): e12012.