Complément cours Planification SMA

- ## Planification à initiative mixte

Définition :

La planification d'une entité autonome (robot, drone, véhicule autonome, …) consiste à construire un plan d'actions pour atteindre un but sans l'intervention d'une entité externe.

La planification à initiative mixte consiste en l'intervention d'une entité externe pour la construction de plan d'actions. Cette intervention se traduit par la spécification des actions à entreprendre lorsque le système se trouve dans certains particuliers. Formellement, un ensemble de couples *(état, action)* est donné au planificateur indiquant les actions à inclure dans le plan pour certains états.

Un planificateur à initiative mixte a, donc, comme entrées *l'état initial, l'état final et mixte* un ensemble de couples (état, action) mixte = {(s,a)} et produit comme sortie un plan d'actions.

Eléments de base pour l'implémentation :

- Soient S_mixte et A_mixte tous les états et actions de mixte.
- Soient A et S les espaces respectifs d'actions et d'états
- I : état initial
- G : état but

Planification_Mixte (I, G, mixte, plan)

```
Si I = G alors retour plan
Sinon
        Si I ∈ S_mixte alors
                plan = plan ∪ {a_mixte | (I, a_mixte) ∈ Mixte}
                Planification_Mixte (progression(I,a_mixte), G, mixte, plan)
        Sinon
                Actions = Activable (I)        /* actions activables à l' état I) */
                Best_action = Meilleur_action (Actions, heuristique)
                Planification_Mixte (progression(I,Best_action), G, mixte, plan ∪
        {Best_action} )
        FINSI
FINSI
```

Remarque : la fonction progression(s,a) permet de calculer le nouvel état s' lorsqu'on applique l'action a dans l'état s.

- Planification jointe

Définition :

La planification jointe étend la planification d'une entité autonome à plusieurs entités. Il s'agit d'un planificateur central qui calcule un plan joint dont les éléments sont les actions jointes de toutes les entités. Plus formellement, soient $\mathbf{A_1}$, $\mathbf{A_2}$, ..., $\mathbf{A_n}$, les ensembles d'actions des entités 1 … n et **S** l'espace d'états de l'environnement dans lequel les entités agissent.
Un état du système composé de plusieurs entités est défini par l'état joint de ces entités ($\mathbf{s_1}$, $\mathbf{s_2}$, ..., $\mathbf{s_n}$) et agit par une action jointe de ces entités ($\mathbf{a_1}$, $\mathbf{a_2}$, ..., $\mathbf{a_n}$,).

Le planificateur calcule des actions jointes ($\mathbf{a_1}$, $\mathbf{a_2}$, ..., $\mathbf{a_n}$,) pour aller d'un état initial **($\mathbf{i_1}$, $\mathbf{i_2}$, ..., $\mathbf{i_n}$,)** à un état but ($\mathbf{g_1}$, $\mathbf{g_2}$, ..., $\mathbf{g_n}$). Tous les agents n'arrivent pas aux états finaux au même moment. Une **fonction heuristique H** doit permettre de planifier que des actions que pour les agents qui n'ont pas encore atteint leurs états buts. Pour les autres une action *nop* doit être appliquée. Pour cela, la fonction **H**($\mathbf{s_1}$, $\mathbf{s_2}$, ..., $\mathbf{s_n}$) renvoie un n-uplet de distances (**$\mathbf{d_1}$, $\mathbf{d_2}$, …, $\mathbf{d_n}$)** où $\mathbf{d_i}$ est la distance **entre l'état $\mathbf{s_i}$ et $\mathbf{g_i}$.** La fonction **H** doit utiliser des critères d'utilité collective (**CUF**) vus en cours sur les fonctions d'utilités collectives pour préférer une action par rapport à une autre.

Squelette de l'algorithme :

Planification_Jointe (**($\mathbf{i_1}$, $\mathbf{i_2}$, ..., $\mathbf{i_n}$,)**, ($\mathbf{g_1}$, $\mathbf{g_2}$, ..., $\mathbf{g_n}$), plan)

Si EGAL (**($\mathbf{i_1}$, $\mathbf{i_2}$, ..., $\mathbf{i_n}$,)**, ($\mathbf{g_1}$, $\mathbf{g_2}$, ..., $\mathbf{g_n}$)) alors retour plan
Sinon
    JointActions = Activable (**($\mathbf{i_1}$, $\mathbf{i_2}$, ..., $\mathbf{i_n}$,)**) */* actions activables à l' état I) */*
    Best_action = Meilleur_action (JointActions, heuristique H)
    Planification_Jointe (progressionJ(**($\mathbf{i_1}$, $\mathbf{i_2}$, ..., $\mathbf{i_n}$,)**,Best_action), **($\mathbf{g_1}$, $\mathbf{g_2}$, ..., $\mathbf{g_n}$)** , plan ∪ {Best_action})
    FINSI
FINSI

- La fonction EGAL teste l'égalité composant par composant.
- La fonction H compare des vecteurs de distance. On peut utiliser les CUF.
- Best_action est une action jointe (($\mathbf{a_1}$, $\mathbf{a_2}$, ..., $\mathbf{a_n}$,)
- La fonction progressionJ applique à chaque état i_k, l'action a_k pour produire le nouvel état i'_k composant du nouvel état joint état joint telle que :

ProgressionJ((**($\mathbf{i_1}$, $\mathbf{i_2}$, ..., $\mathbf{i_n}$,)**,), (($\mathbf{a_1}$, $\mathbf{a_2}$, ..., $\mathbf{a_n}$,)) = (progression(**$\mathbf{i_1}$,$\mathbf{a_1}$**), ..., progression(**$\mathbf{i_n}$,$\mathbf{a_n}$**))

- Planification coopérative distribuée

Définition :

La planification distribuée consiste en un ensemble de planification locale à chaque entité (agent). Chaque agent construit son plan local pour aller de l'état initial à l'état final. Les plans locaux doivent ainsi être coordonnées pour éviter les inconsistances (un robot veut ouvrir la porte et un autre veut la fermer) et les conflits (deux robots veulent sortir par la même porte. Plusieurs approches :

- **Planification avant coordination** : cette approche consiste à permettre à chaque agent K de construire un plan local P_k puis le coordonner avec les plans locaux ds autres agents $P_{i\#k}$. On trouve donc deux phases :
  - Planification locale : chaque agent calcule par un algorithme de planification un plan local. Les agents s'échangent les plans locaux pour obtenir tous les plans { P_1, ..., P_n }
  - Chaque agent vérifie la consistance de son plan avec les autres.
  - Si le plan est consistant, il le retourne
  - Sinon, des agents choisis (Heuristique ou Vote), changeront leurs plans jusqu'à aboutir à une consistance.
- **Planification pendant/après coordination** : cette approche définit toutes les actions jointes inconsistantes et propage les entre les agents. Chaque agent à chaque étape de planification doit échanger les actions planifiées et vérifient leur consistance.
  - Etape de propagation : échange des actions planifiées
  - Vérifications de la consistance : Si problème un agent ou plusieurs (heuristique ou vote) change son action et la propage aux autres agents.
  - Ce processus est répété jusqu'à ce que tous les agents construisent leurs plans consistants.

La planification avant coordination est une approche qui fait de la planification totale suivie de coordination alors que pendant ou après il s'agit de planification partielle suivie de coordination partielle.

Planification distribuée de tâches :

Dans le support de cours, un exemple de logistique a été introduit pour le transport de marchandises entre villes (qui peuvent être des villes de pays différents). Les tâches sont le transport d'une ville à l'autre d'un même pays et le transport entre deux aéroports. Chaque pays a un agent qui assure les transports entre ses villes et un agent de transport aérien. Une mission M1 de transport d'une ville A vers une ville B peut se traduire par un transport de la ville A à l'aéroport du même pays, puis un transport vers l'aéroport du pays de la ville B, puis de l'aéroport à la ville B. On voit bien ici trois agents impliqués. Mais l'agent aéroport ne peut assurer sa tâche si l'agent du pays A n'a pas réalisé la sienne. **On parle alors de contrainte de précédence entre tâches**. Si on se trouve avec un ensemble de Mission, chaque agent doit assurer plusieurs tâches et tenir compte des tâches des autres en respectant les contraintes de précédence. Plus formellement, chaque tâche d'un agent représente une étape du plan (un nœud) et chaque tâche a des contraintes de précédences (des arcs). Cela permet de construire un **graphe de dépendance** entre agents signifiant qu'un agent doit faire avant ou après une tâche qu'un autre.

L'objectif est de calculer des contraintes de précédentes entre les tâches d'un seul agent (des arcs entre ces nœuds) de sorte que le graphe de précédence reste cohérent. **Le graphe est dit cohérent s'il n'y a pas de cycle**.

- **Approche** : Consiste à estampiller les nœuds pour leur donner un ordre dans le plan local tels que tous les plans locaux respectent le graphe de précédence.
    - **Etape initialisation** : estampiller avec le label 1 tous les nœuds qui n'ont pas d'arc rentrant.
    - **Propagation** : propager les labels en utilisant le graphe de précédence en estampiller le nœud d'arrivée au label reçu +1. Répéter la propagation partout.
    - **Vérification de la consistance** : chaque agent doit vérifier que les labels de ses nœuds sont dans un ordre total, sinon en créant un ordre total puis propager les labels modifiés. Par exemple, si un agent a trois nœuds labélisés (1 3 3), il peut le changer en (1 4 3) et propager le label 4 aux suivants. Si l'agent converge vers cet ordre cela signifie que la tâche 1 va être faite à l'étape 1, à l'étape 2 rien, l'étape 3 la tâche 3 et à l'étape 4 la tâche 2.

Travail à faire :

- Implémenter les algorithmes suivants :
    a. **Planification_mixte** : on se donne un robot qui se déplace dans une grille avec obstacle (placé aléatoirement) pour aller d'une position A à une position B et une liste d'actions donnée par un opérateur.
    b. **Planification_jointe** : deux robots se déplacent dans une grille chacun à un état initial et un état but. On calcule un plan joint pour que chacun rejoint sa destination sans rentrer en collision et en évitant les obstacles placés aléatoirement dans la grille.
    c. **Planification distribuée de tâches** en utilisant l'exemple de logistique donné en cours.