IFT 608 / IFT 702
Planification en intelligence artificielle

Planification multi-agents

Froduald Kabanza
Département d'informatique
Université de Sherbrooke

Sujets couverts

- Types de problèmes de planification multi-agents
- Planification coopérative
- Planification contre des adversaires et la théorie des jeux

Références:

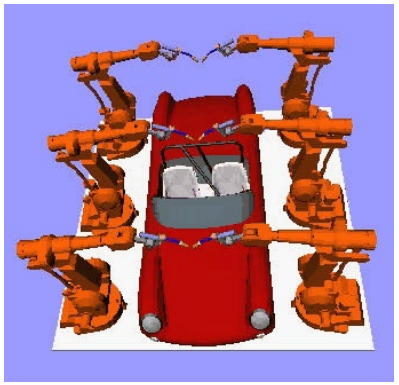
Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. *Chapitres 1 à 2 et Chapitre 5.*

Browne *et al.* A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games, VOL. 4, NO. 1, March 2012*

+ Lectures suggérées à la fin

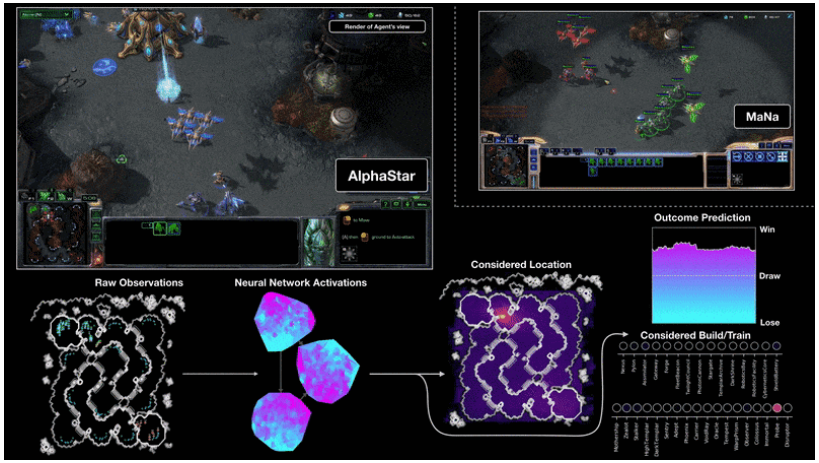
TYPES DE PROBLÈMES

Planification multi-agents



Planification par ou pour plusieurs agents

- **Coopératif** : but commun, fonction d'utilité commune.
  - Plusieurs robots coopératifs

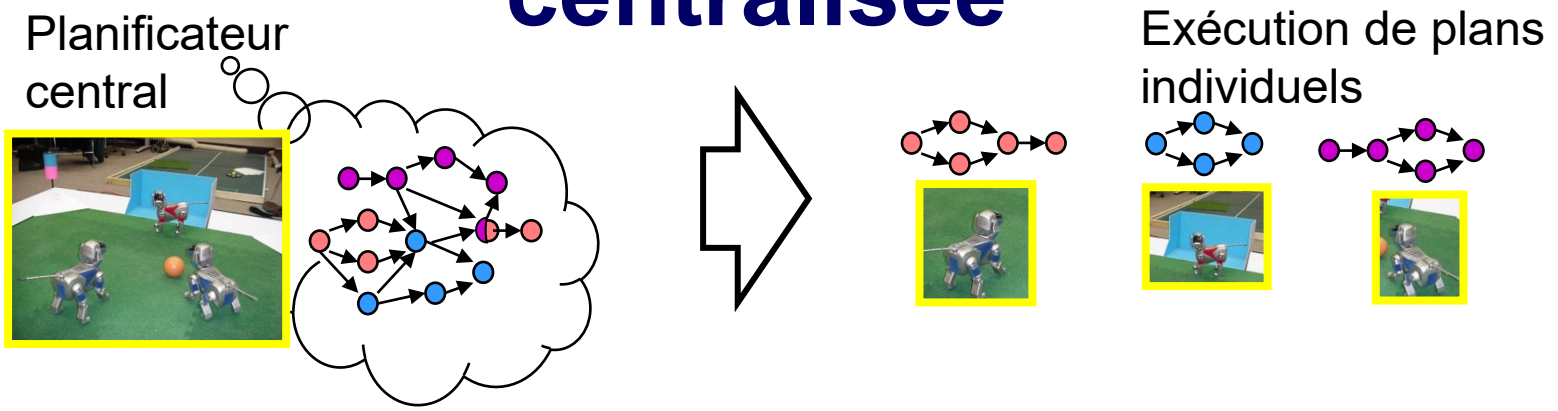


- **Compétitif** : buts opposés
  - Jeux de stratégie temps réel (RTS)
  - Opérations militaires

- **Coalition**: coopératif et compétitif
  - Jeux RTS multi-joueurs
  - Jeux d'équipe comme le soccer

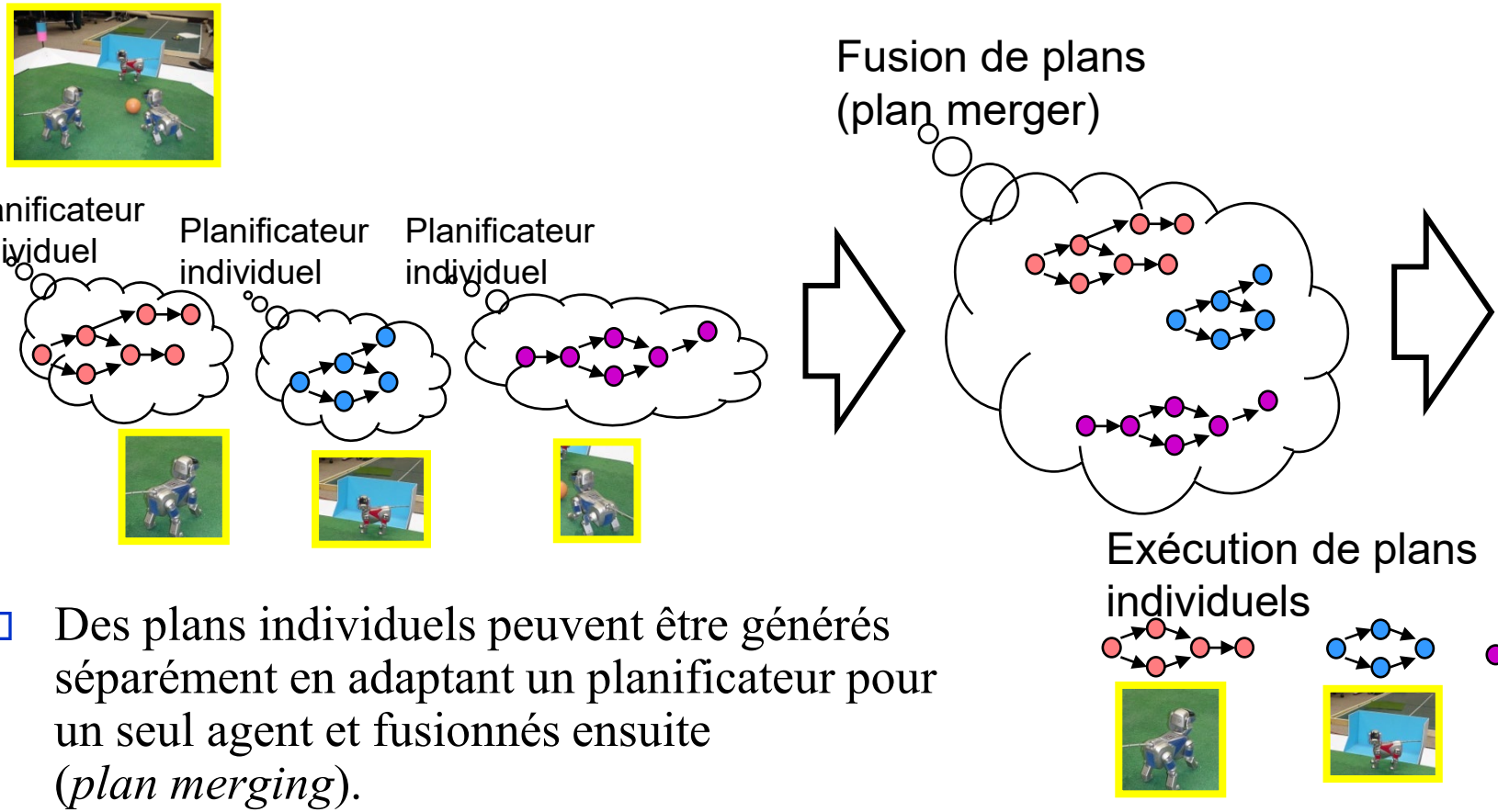
PLANIFICATION COOPÉRATIVE

Planification coopérative centralisée



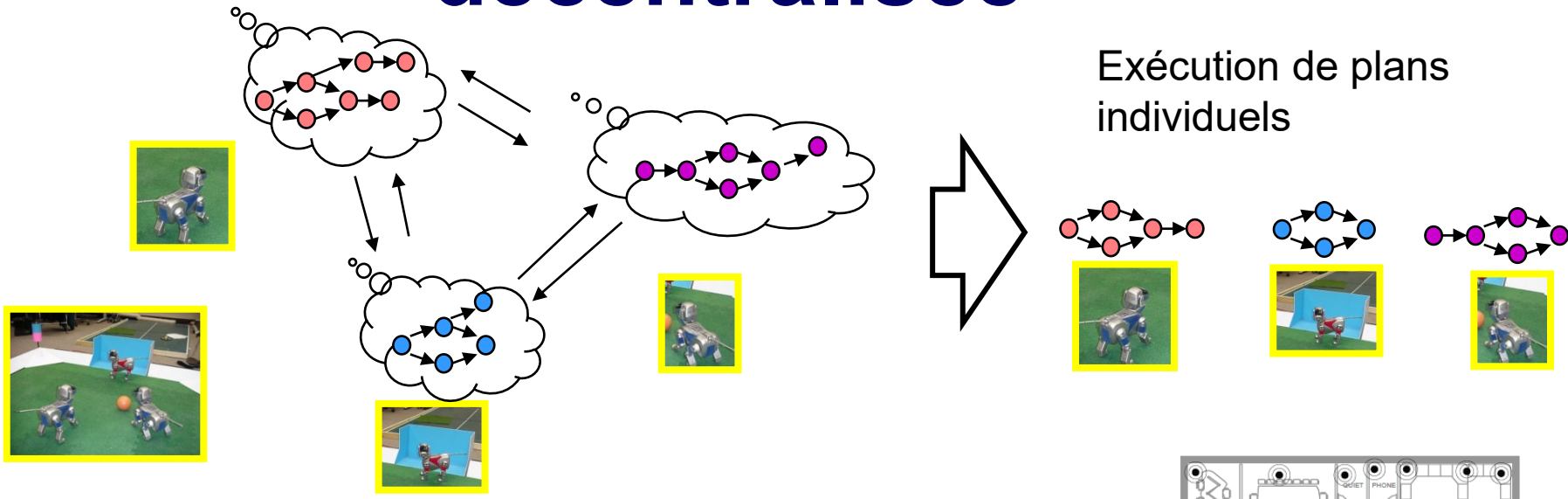
- Planification centralisée, exécution décentralisé
  - Le planificateur central doit être capable de traiter des
    - » Activités concurrentes
    - » Buts temporels
  - Il faut un middleware assurant la communication entre les agents (par exemple: ROS, DDS).
- **Avantage**: Généralisation facile des algorithmes usuels de planification pour un seul agent.
- **Désavantage**: généralement inefficace

Planification coopérative décentralisée avec fusion de plans



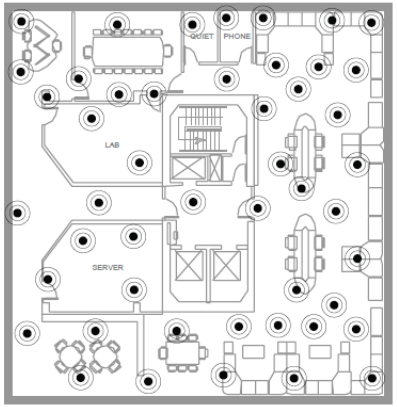
- Des plans individuels peuvent être générés séparément en adaptant un planificateur pour un seul agent et fusionnés ensuite (*plan merging*).
- La planification par recherche dans l'espace de plan vu antérieurement s'apprêtent particulièrement bien à la fusion des plans.

Planification complètement décentralisée



- Dans une approche complètement décentralisée, chaque agent calcule son plan en échangent des données avec les autres.
- Une des approches est « Distributed CSP (DCSP) ». C'est une généralisation des problèmes CSP vus dans IFT615.

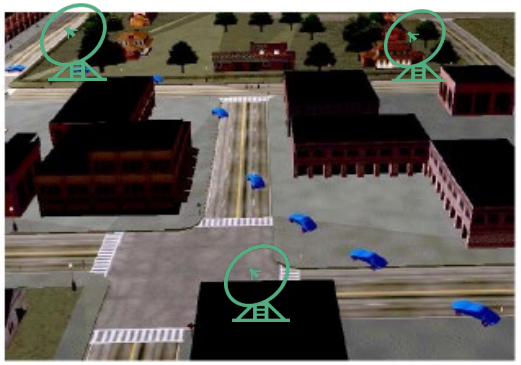
Chapitre 1 & 2 de [Shoam et Leyton-Brown]



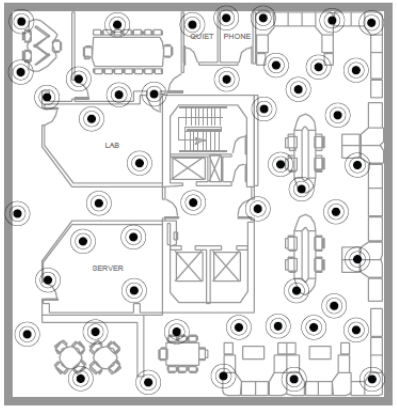
Indor sensor network

Distributed CSP

- Une généralisation à plusieurs agents de backtracking-search pour CSP (IFT615)
- Pour des problèmes de planification coopérative
  - Exemple: Réseau de capteurs distribués (SensorDCSP)
- Chapitre 1 de [Shoam et Leyton-Brown]



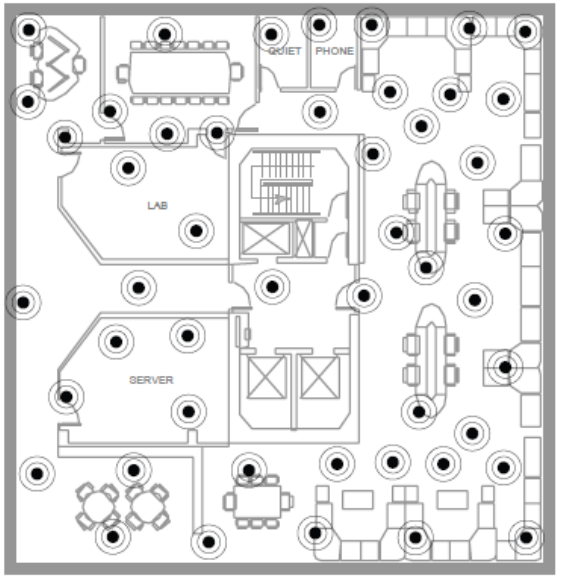
Outdoor sensor network



Indor sensor network

Sensor DCSP - Enoncé du problème

- Plusieurs capteurs: *s1, …, sm*
  - Chaque capteur a un rayon d'action
  - Peut être obstrué par des obstacles dans l'environnement
  - Peut fonctionner sur des fréquences différentes
  - Les rayons d'action des capteurs peuvent se chevaucher
- Plusieurs cibles à suivre: *t1, …, tn*.
- Problème: Allouer des capteurs aux cibles, de sorte que l'on puisse suivre les cibles en tout temps et qu'il n'y ait pas d'interférences entre les capteurs.
  - Il y a interférence lorsque deux capteurs avec des rayons d'action qui se chevauchent utilisent la même fréquence.
- Peut se modéliser comme un problème CSP

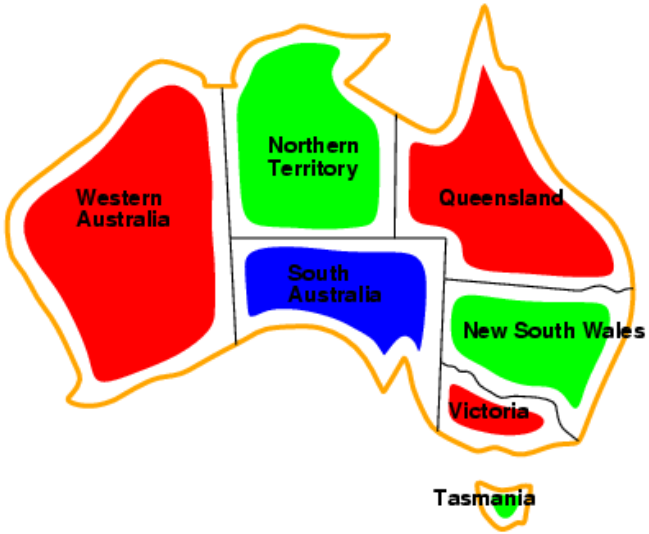


Rappel IFT615 – Problème CSP

- Formellement, *un problème de satisfaction de contraintes* (ou *CSP* pour *Constraint Satisfaction Problem*) est défini par:
  - Un ensemble fini de *variables* $X_1, \ldots, X_n$.
    - » Chaque variable X_i a un *domaine* D_i de *valeurs* permises.
  - Un ensemble fini de *contraintes* $C_1, \ldots, C_m$ sur les variables.
    - » Une contrainte restreint les valeurs pour un sous-ensemble de variables.
- Un *état d'un problème CSP* est défini par une *assignation* de valeurs à certaines variables ou à toutes les variables.
  - $\{X_i=v_i, X_n=v_1, \ldots\}$.
- Une assignation qui ne viole aucune contrainte est dite *consistante* ou *légale*.
- Une *assignation* est *complète* si elle concerne toutes les variables.
- Une *solution* à un problème CSP est une *assignation complète* et *consistante*.
- Parfois, la solution doit en plus *maximiser une fonction objective* donnée.

Rappel IFT615 - Exemple : Colorier une carte

- On vous donne une carte de l'Australie :

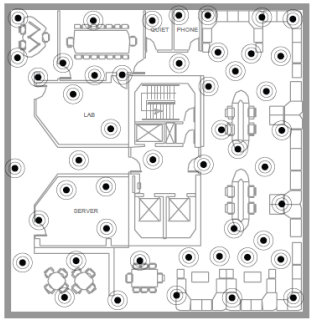


- Et on vous demande d'utiliser seulement trois couleurs (*rouge*, *vert* et *bleu*) de sorte que deux états frontaliers n'aient jamais les mêmes couleurs.
- On peut facilement trouver une solution à ce problème en le formulant comme un problème CSP et en utilisant des algorithmes généraux pour CSP.

Rappel IFT615 - Backtracking search

```
function BACKTRACKING-SEARCH(csp) return a solution or failure
    return BACKTRACK({} , csp)
function BACKTRACK(assignment, csp) return a solution or failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(var, assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment then
            add {var=value} to assignment
            inferences ← INFERENCES(csp, var, value) // e.g., AC-3
            if inferences ≠ failure then
                add inferences to assignment
                result ← BACTRACK (assignment, csp)
                if result ≠ failure  then return result
        remove {var=value} and inferences from assignment
    return failure
```

DCSP



- Dans DCSP, chaque variable est gérée par un agent.
- Le problème demeure de trouver une assignation qui satisfait les contraintes.
- Chaque agent décide de la valeur de sa variable avec une certaine autonomie.
- Les agents se coordonnent en parllèle.

- Chaque agent n'a pas une vue globale des assignations, mais il peut communiquer avec les agents voisins (selon le graphe des contraintes) pour connaître leurs valeurs.

- Un algorithme DCSP consiste à avoir les agents qui communiquent avec leurs voisins, chacun mettant à jour sa valeur, de sorte que le processus converge éventuellement vers une assignation complète et satisfaisante.

- Le chapitre 1 [Shoam et Leyton-Brown] décrit deux algorithmes en détail.

PLANIFICATION ADVERSARIALE

Concepts basiques

- Comme dans la planification MDP, le but est pour chaque agent de **maximiser sa fonction d'utilité**.
  - La différente notable est que nous avons maintenant plusieurs agents (deux dans l'exemple) possiblement avec des fonctions d'utilité opposées.
- L'équivalent d'un plan ou politique dans les jeux est **une stratégie**
  - Choix d'actions pour un jouer pour toutes les phases du jeu, autrement dit pour chaque état.
- **Stratégie pure** (fixée pour chaque état du jeu) vs **stratégie mixte** (aléatoire)
- Au lieu de plan/stratégie optimal, on parle de **concept d'équilibre**.
- Il y a plusieurs concepts d'équilibre.

Théorie des jeux

- Types de jeux
  - Coopératif / Non coopératif
  - Somme nulle / Somme générale
  - Simultanés / Séquentiels
  - Information complète / incomplète
  - Information parfaite / imparfaite

- Types de représentation
  - Normale (matricielle)
  - Extensive (arbre de jeu)

Équilibre de Nash

- Étant donné un ensemble de stratégies (une pour chaque agent), elles sont en équilibre de Nash si et seulement si chaque stratégie est la meilleure réponse face aux autres stratégies

– Autrement dit, aucun agent n'a intérêt à dévier de sa stratégie si les stratégies des adversaires restent fixes.

– L'équilibre de Nash est conservatrice

  - Donne une stratégie optimale si effectivement les autres agents jouent selon l'équilibre (jouent de façon optimale)

  - Elle n'exploite pas les faiblesses éventuelles des autres agents

Approches algorithmiques en général

- Jeu matriciel à 2 joueurs à somme nulle
  - Problème d'optimisation linéaire
- Jeu matriciel à 2 joueurs à somme générale
  - Problème d'optimisation quadratique
  - Algorithme de Lemke-Howson
  - Algorithme d'énumération des supports (*Support enumeration*)
- Jeu matriciel à *n* joueurs
  - *Govindan and Wilson's continuation method*
- Jeux extensifs (représentation sous-forme d'arbres de jeu)
  - Minmax (jeu séquentiel à information parfaite)
  - Élagage (*pruning*) de stratégies strictement dominées
  - Expectimax (jeu à information incomplète)
  - Échantillonnage *Monte Carlo*
  - Recherche heuristique

Résumé

- La planification multi-agents concerne la planification pour plusieurs agents.
- La théorie des jeux fournit les concepts de solution (équilibres) et les algorithmes de prise de décision.
- L'intelligence artificielle distribuées fournit des méthodes de coordination centralisées, décentralisées, ou hybrides.
- Pour la planification coopérative, les approches de planification par recherche dans l'espace de plans, combinées avec DCSP offrent un cadre de résolution fréquemment rencontré dans la littérature.
- **Pour l'examen**: retenir les différents concepts (définitions), types de problèmes et types d'approches de solutions correspondantes.

Articles suggérés

- Arulkumaramn et al. AlphaStar: An Evolutionary Computation Perspective. arXiv:1902.01724, 2019.
- Espeholt et al. MPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. arXiv:1802.01561v3 [cs.LG] 28 Jun 2018.
- Jaderberg et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. arXiv:1807.01281 [cs.AI]
- Lanctot et al. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. arXiv:1711.00832v2 [cs.AI] 7 Nov 2017.