# Intelligence Artificielle Distribuée
# Systèmes Multi-Agents

Guillaume Hutzler
LaMI (Laboratoire de Méthodes Informatiques)
SyDRA (Systèmes Distribués, Réactifs et Adaptatifs)
hutzler@lami.univ-evry.fr
http://www-poleia.lip6.fr/~hutzler

## Plan du cours

- Introduction
- Techniques multi-agents et applications
  - Eco-Problem Solving
  - Ant Colony Optimization
  - Robocup
- Conclusion

## Eco-Résolution (1)

- L'éco-résolution (Eco Problem Solving - EPS) est un framework de SMA réactif dédié à la résolution de problèmes.
  - existe en : Lisp, Smalltalk, C++, Java
- Il s'appuie sur :
  - une décomposition structurelle du problème (i.e. "objet")
  - un modèle d'agent séquençant trois comportements basiques (satisfaction, agression, fuite)
  - une description explicite de l'état initial et de l'état final du problème.

## Eco-Résolution (2): le modèle d'agent

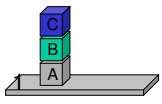
- Chaque agent a
  - un but = l'agent qui définit sa position finale
  - des accointances = les agents qu'il connaît
  - des gêneurs = les agents qui le gênent pour atteindre son but
  - des dépendances = les agents qui dépendent de lui pour pouvoir se satisfaire
- Le modèle de comportement est un automate à états finis (différences entre les versions):
  - satisfait: l'agent ne fait rien
  - rechercheSatisfaction: l'agent cherche à se satisfaire
  - rechercheFuite: l'agent a été agressé et tente de fuir
  - fuite: l'agent fuit

## Eco-Résolution (3)

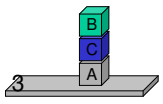
- Informellement...
  - Si le but d'un agent n'est pas satisfait, l'agent demande à son but de se satisfaire et se place dans ses dépendances.
  - Quand un agent se satisfait, il informe ses dépendances qu'elles peuvent se satisfaire.
  - Quand un agent ne peut se satisfaire, il recherche les gêneurs parmi ses accointances et les agresse (leur demande de fuir).
  - Quand un agent cherche à fuir, il recherche les gêneurs parmi ses accointances et les agresse.
- La résolution du problème est:
  - incrémentale
  - réactive

## EPS: Exemple des cubes (1)

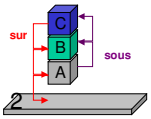
- Situation initiale



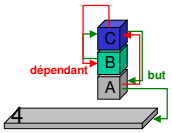

- Situation finale



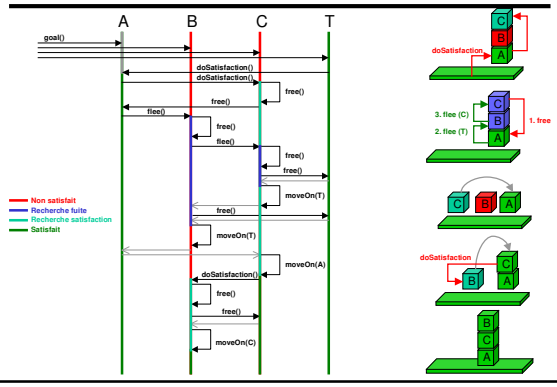

- Initialisation des accointances




- Initialisation des buts

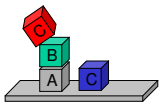


## EPS: Exemple des cubes (2)



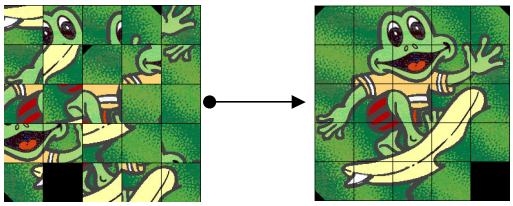

## EPS: Exemple des cubes (3)

- Convergence étudiée par [Jacopin 92]
- Pas d'état du problème, pas de backward chaining.
- Solution non optimale, mais :
  - Ajout de contraintes (table réduite) : modification de la méthode chercherPlacePourFuir(contrainte) [Drogoul 91]
  - Résolution dynamique : problème du "misachieving baby" [Bura & al. 92]




## Le problème du Taquin

- Problème fortement combinatoire
  - Problème le plus populaire en IA, après les échecs
  - Solution optimale pour le 5x5, impossible au-delà



## Taquin : Approche classique (1)
### Résolution de problème type IA

- Parcours d'un espace d'états
  - Un état = la position de chacun des palets
  - Transition = passage d'un état à un autre
    - 2, 3 ou 4 nouveaux états peuvent être atteints suivant la position du blanc (coin, bord, centre)
    - définit un arbre (avec entre 2 et 4 fils pour chaque nœud)
    - le nombre de feuilles croit exponentiellement avec le nombre de transitions

Algorithmes de type A* ou AO*

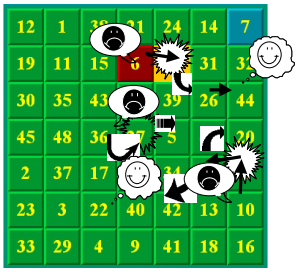
- parcours systématique de l'arbre avec attribution de scores aux différents états
- exploration prioritaire des états avec les meilleurs scores
- élimination des branches dont on sait avec certitude qu'elle ne peuvent pas donner de meilleur résultat qu'une solution déjà obtenue

## Taquin : Approche classique (2)
### Problèmes liés à l'approche classique

- Solution optimale
  - espace de recherche énorme exploré systématiquement
  - quand on explore une solution qui n'est pas intéressante, retour arrière pour explorer d'autres solutions
    - coûteux en temps d'exécution et en mémoire
- Solution approchée
  - utilisation d'heuristiques pour limiter l'exploration
    - calcul d'une distance entre un palet et son but (distance de Manhattan)
    - choix de l'état dans lequel la somme des distances des palets à leur but est la plus faible
  - Etudié par Korf : adaptations de A* (RTA*, LRTA*)
- Pb quand perturbations extérieures
  - Nécessaire de tout recommencer avec un nouvel état initial

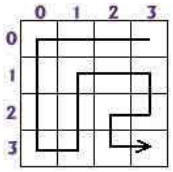
## Taquin : Solution EPS (1)

- Le problème est décomposé en deux types d'éco-agents
  - les cases (patchs)
  - les palets
- Chaque palet a pour but une case, comme accointances les cases adjacentes, comme "gêneurs" les palets posés dessus



## Taquin : Solution EPS (2)

- La résolution est séquentielle dans le cas du taquin "normal" (un seul espace de liberté)
  - ordonnancement des agents pour la résolution
  - Initialisation des buts et des dépendances


## Taquin : Solution EPS (3)

- Introduction de mécanismes de blocage
  - blocage temporaire des patchs sur lesquels se trouvent des palets satisfaits
  - blocage temporaire du patch sur lequel se trouve l'agresseur
    - but = éviter les agressions mutuelles sans fin
    - blocage levé si l'agent agressé n'a pas d'autre choix
  - blocage définitif des lignes et colonnes satisfaites

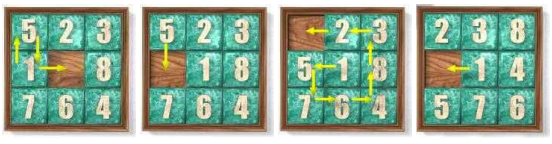
## Taquin : Solution EPS (4)

- Heuristiques pour le choix d'une place
  - pour s'échapper :
    - choix parmi les patchs adjacents, de celui qui est le plus proche du blanc
    - prise en compte des patchs bloqués
    - Possibilité de s'échapper sur le but
  - pour se satisfaire
    - choix parmi les patchs adjacents, de celui qui est le plus proche du but
    - si plusieurs équivalents, choix de celui qui est le plus proche du blanc
    - prise en compte des patchs bloqués

## Taquin : Solution EPS (5)

- Transmission de contraintes
  - attaquer pour se satisfaire
    - si l'attaquant n'est pas adjacent à son but, l'attaquant donne comme contrainte à l'agent agressé de ne pas s'échapper sur le but
    - si le but est adjacent, la contrainte donnée correspond à un ordre choisi entre les patchs
  - attaquer pour s'échapper
    - la contrainte est le patch sur lequel le palet à choisi de fuir (donc celui sur lequel se trouve l'agent attaqué)

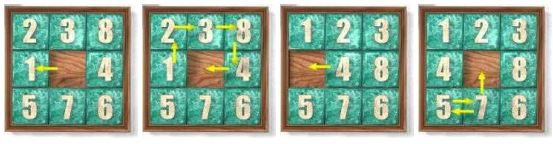
## Taquin : Exemple 1.a



(a) (b) (c) (d)

- (a)
  - 5 agresse 1 en retour car 2 est bloqué
- (c)
  - 5 ne fuit pas vers le blanc car c'est le but de son agresseur 1
  - 2 s'enfuit vers le blanc car 1 et 3 bloqués

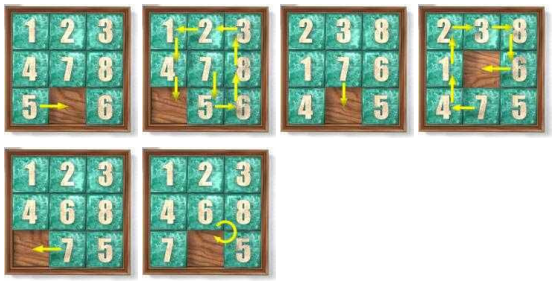
## Taquin : Exemple 1.b



(a) (b) (c) (d)

- (b)
  - 3 agresse 8 plutôt que de fuir sur le blanc car c'est son but et il n'est pas bloqué
- (d)
  - Recherche de satisfaction de 7 comme pour 1

## Taquin : Exemple 1.c

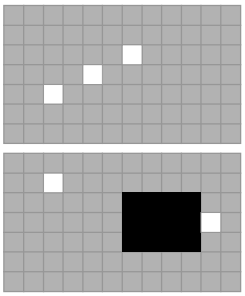


- Une fois 7 satisfait, la colonne 0 et la ligne 0 sont bloqués

## Taquin : Exemple 2

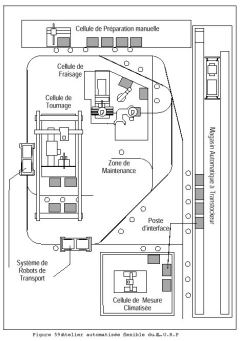
## Taquin : résultats obtenus avec l'EPS

- Complétude et décidabilité démontrées dans [Drogoul,Dubreuil 93]
- Résultats expérimentaux obtenus jusqu'à des taquins de 1000x1000...
- Complexité proche de la complexité approximée de la solution optimale
- Adaptation à des instances de la même classe de problèmes: taquins à trous, irréguliers, etc.



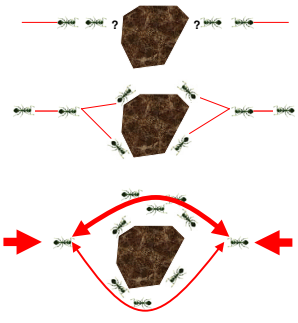
## EPS : autres applis

- C. Sohier (ENS- Cachan)
  - Ordonnancement automatisé d'un atelier flexible
  - Chaque machine-outil, robot, outil, etc. est représenté par un éco-agent.
- K. Ghedira (Univ. Tunis III)
  - Satisfaction de contraintes
  - *flow shop* scheduling
  - recuit simulé distribué
- Autres
  - problèmes "jouets" (n-reines, tours de Hanoï, etc.)
  - projet Carosse (PSA)



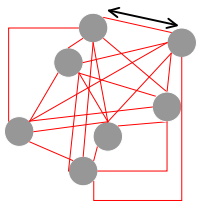

## ANT – Ant Colony Optimization (ACO)

- Système de résolutions de problèmes combinatoires basés sur une forte analogie avec les mécanismes mis en œuvre dans les colonnes de fourmis [Colorni - Dorigo 91]
- Les fourmis (cf. figure) se déplacent en déposant des phéromones et sont attirées par elles.
- Décision collective du plus court chemin.



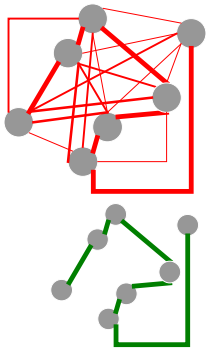
## ANT : Voyageur de commerce (TSP) (1)

- TSP: Un ensemble de villes reliées entre elles par des arcs définissant une certaine distance.
- Le problème est de trouver le chemin minimal permettant de ne parcourir chaque ville qu'une fois.
- Les TSP peuvent être symétriques ou asymétriques.



## ANT - TSP (2) : description informelle

- Chaque arc se voit attribuer, en plus de sa distance, un taux de phéromones.
- Les fourmis font un tour complet en choisissant les villes selon une règle probabiliste : emprunter les arcs les plus courts et les plus riches en phéromones.
- A chaque itération, une règle globale fait (1) s'évaporer une partie des phéromones et (2) augmente le taux de chaque arc visité en fonction de la longueur des parcours.



## ANT - TSP (3) : description précise

- $N$ ensemble de $n$ noeuds, et $N_i$ l'ensemble des voisins du noeud $i$

  $E$ ensemble des arcs qui les connectent

  $d_{ij}$ longueur de l'arc $(i,j) \in E$ , et $\eta_{ij} = 1/d_{ij}$, sa valeur heuristique

  $F$ ensemble de $m$ fourmis
- $t$ (compris entre $0$ et $tmax$) identifie l'itération en cours
- $\tau_{ij}(t)$ est le taux de phéromones associé à l'arc $(i,j)$
- $M_k \subset N$ est une mémoire associée à la fourmi k (qui lui permet de savoir quels noeuds elle a parcouru, et lesquels elle ne doit plus parcourir). Chaque noeud visité y est ajouté.

## ANT - TSP (4) : description précise

- A chaque noeud $i$, la fourmi construit une table de décision $A_i$ contenant les éléments suivants :

$$a_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}} \forall j \in N_i$$

où $\alpha$ et $\beta$ représentent les poids relatifs de la trace de phéromones et de la valeur heuristique

- La probabilité avec laquelle elle choisit de se déplacer de $i$ à $j$ à l'itération $t$ est

$$p_{ij}(t) = \frac{a_{ij}(t)}{\sum_{l \in N_i} a_{il}(t)}$$

## ANT - TSP (5) : description précise

- Quand toutes les fourmis ont complété leur tour, chacune dépose sur les arcs qu'elle a visité une quantité de phéromones :

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} 1/L^{k}(t) & si\ (i,j) \in T^{k}(t) \\ 0 & si\ (i,j) \notin T^{k}(t) \end{cases}$$

où $T^k(t)$ représente le chemin parcouru par la fourmi, et $L^k(t)$ sa longueur

- Puis les phéromones sont décrémentées selon la règle :

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t)$$

où $\rho \in [0,1]$ représente le coefficient d'évaporation

## ANT - TSP (6) : description précise

- Les paramètres qui pilotent l'algorithme sont donc :
  - $\tau_{ij}(0)$, le taux initial de phéromones, qu'il faut initialiser
  - $m$, le nombre de fourmis considérées
  - $\alpha$ et $\beta$, les valeurs contrôlant les poids respectifs des phéromones et de l'heuristique
  - $\rho$, le coefficient d'évaporation
- En pratique, les meilleurs résultats ont été obtenus avec les valeurs suivantes :
  - $\tau_{ij}(0) = 0,1$
  - $m = |N|$ (nombre de noeuds)
  - $\alpha = 1$ et $\beta = 5$,
  - $\rho = 0,5$

## ANT - TSP (7) : amélioration

- ACS (95) : amélioration de l'algorithme
  - les fourmis sont positionnées aléatoirement
  - elles modifient localement le taux de phéromone des arcs empruntés
  - seul le plus court chemin est récompensé par la règle globale
  - proche de l'apprentissage par renforcement

## ANT - TSP (8) : résultats

- Comparaison avec d'autres méthodes
  - à défaut de connaître l'optimum, donne des résultats équivalents voire meilleurs par rapport aux autres algorithmes (algos génétiques, cartes auto-organisatrices, programmation évolutionniste, etc.) pour la plupart des jeux d'essais
- Qualités
  - bonnes performances pour les pbs statiques
  - pas de difficultés avec des pbs dynamiques
  - Flexibilité et robustesse
- Défauts
  - moins performants avec des pbs générés uniformément aléatoirement (mais peu courants dans la réalité)

## Routage de paquets IP : ACRouting

- Routage
  - acheminement des paquets IP depuis la source vers la destination à travers des nœuds de routage (switchs)
  - utilisation de tables de routage au niveau des switchs pour décider où les paquets doivent être redirigés en fonction de leur destination
- Ant Colony Routing
  - des agents fourmis qui parcourent le réseau renforcent plus ou moins les entrées de la table de routage en fonction du temps qu'ils ont mis pour acheminer un message d'un point à un autre
  - un mécanisme d'évaporation rafraîchit régulièrement les entrées de manière à prendre en compte dynamiquement les conditions de trafic changeantes

## Autres applications de l'ACO

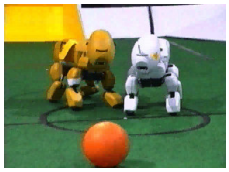
- Beaucoup d'applications de type "recherche opérationnelle"
  - coloration de graphes (1997 ANTCOL)
  - superséquence commune la plus courte (1999 AS-SCS)
  - assignement quadratique (1999 HAS-QAP, MMAS-QAP, AS-QAP)
  - ordonnancement de tâches (1999 ACS-SMTTP)
  - routage de véhicules (1999 AS-VRP, MACS-VRPTW)
  - attribution de fréquences (2000 ANTS-FAP)
  - Ordonnancement séquentiel (1997 HAS-SOP)

## ANT-ACO: Conclusion

- Les ACO sont une classe d'algorithmes bien adaptés aux problèmes combinatoires qui peuvent s'exprimer sous forme de parcours.
- Mais ils partagent avec l'EPS un certain nombre de points critiques :
  - espace de paramètres très important
  - convergence difficile à prouver
  - heuristiques locales difficilement généralisables
  - méthodologie inexistante (sauf pour EPS: méthodologie objet)

## RoboCup

- Construire des groupes de robots capables de jouer à un sport collectif (ressemblant au football)
- Différentes catégories
  - small size
  - middle size
  - simulation
  - legged robots



## Intérêts de ce benchmark

- Représentatif des contraintes qui pèsent sur la conception de systèmes multi-agents
  - action & perception locales
  - pas de coordination globale
  - nécessité de coopération
- Confronte des techniques essentiellement logicielles à un environnement dynamique réel
  - Imperfection de la perception et de la communication
  - Incomplétude de l'information
  - Hiatus entre intention (prévision) et action
  - Possibilités de pannes

## Caractéristiques nécessaires

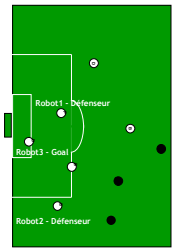
- Fonctionnement collectif
  - tolérance aux pannes individuelles
  - Coopération (explicite / implicite)
  - Gestion souple des rapports actions/tactiques/stratégies
- Fonctionnement individuel
  - perceptions et actions fortement couplées
  - Adaptativité au contexte
- (plus complexe) Reconnaissance des intentions
  - Modélisation des autres agents
  - Modélisation des intentions
  - Reconnaissance de stratégies

## Choix d'un mode d'organisation

- Buts collectifs : marquer le plus de buts, en encaisser le moins possible
  - Sous-buts individuels : contrôler la balle, assurer la défense d'une zone, marquer un adversaire, etc.
- Problème : Comment "distribuer" les buts collectifs de façon adéquate ?
- Plusieurs solutions :
  - organisation statique (chaque joueur se voit confier un rôle fixe)
  - organisation dynamique (chaque joueur prend un rôle selon le contexte dans lequel il se trouve)
  - auto-organisation (pas de rôles différenciés : les joueurs ont tous le même modèle de comportement)

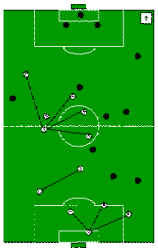
## Organisation statique

- **Intérêts**
  - Distribution établie une fois pour toutes
  - Programmation simple des agents (leur rôle est fixe)
  - Programmation simple de stratégies (groupes d'agents prédéterminés)
- **Difficultés**
  - Reconnaissance de la stratégie en cours
  - Reconnaissance du contexte global
  - Aucune tolérance aux pannes (et si le goal tombe en panne ?)
  - Nécessite une importante communication entre agents (compromis à effectuer avec les actions)



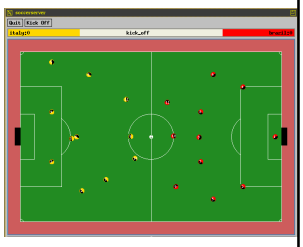

## Organisation dynamique

- **Intérêts**
  - Programmation "homogène" des agents
  - Bonne tolérance aux pannes individuelles
  - Programmation abstraite de stratégies (enchaînements de rôles, et pas d'agents)
- **Difficultés**
  - Reconnaissance de la stratégie en cours
  - Conflits de rôles (évaluation du contexte)
  - Choix des stratégies de groupe et conflits de stratégies
  - Nécessite une importante communication entre agents



## Auto-organisation

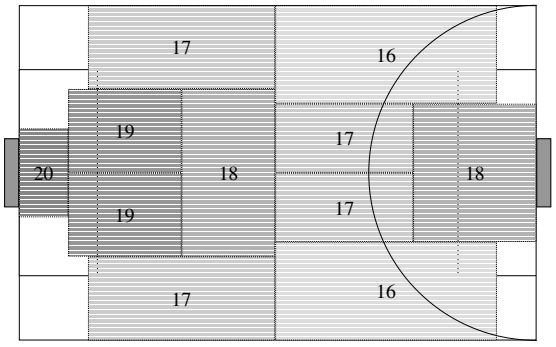
- **Intérêts**
  - Programmation identique des agents
  - Tolérance aux pannes et robustesse
- **Difficultés**
  - Les stratégies ne sont "qu'émergentes"
  - Connaissances empiriques (expérimentations successives)
  - L'environnement doit être "préparé"



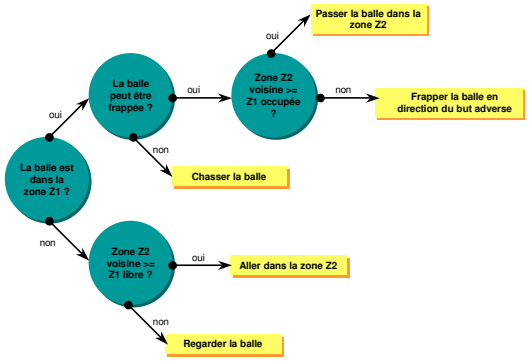
## Exemple (1) : Hypothèses

- Hétérogénéité environnementale
  - découpe du terrain en zones d'importances inégales
- Homogénéité comportementale
  - pas de spécialisation, comportement stéréotypé
- Pas de communication
  - Le placement sur le terrain est une forme de communication
- Pas de mémorisation
  - Absence de modèles des autres
- Pas de planification
  - Ensemble d'actions réflexes
- Ensemble restreint de comportements
  - Aller dans une zone, regarder la balle, chasser la balle, frapper la balle

## Exemple (2): Découpe possible du terrain




## Exemple (3): modèle de comportement simple




## Exemple (4): jeu de passes