Systèmes Multi-Agents « réactifs »

Guillaume Hutzler
IBISC (Informatique Biologie Intégrative et Systèmes Complexes)
LIS (Langage Interaction et Simulation)
prenom.nom@ibisc.univ-evry.fr
http://www.ibisc.univ-evry.fr/~hutzler/Cours/SMA.html

Philosophie (1)

- Hypothèse d'inscription physique
  - les représentations d'un système intelligent doivent s'inscrire dans le monde physique
  - L'intelligence 'réelle' est située dans le monde, pas dans des systèmes désincarnés comme des systèmes experts
  - un comportement 'intelligent' (= organisé) peut émerger des interactions des agents avec leur environnement

Philosophie (2)

- Idées principales
  - un agent ne possède pas de représentation explicite (ni d'eux-mêmes, ni des autres, ni de leur environnement);
  - un agent n'effectue pas de raisonnement abstrait (nécessite des représentations);
  - l'intelligence (l'organisation) est une propriété émergente; l'organisation des agents entre eux n'est qu'un effet induit de leur activité et de leurs interactions : elle n'est explicitée à aucun niveau dans le système, ce n'est pas une donnée du problème;

Pourquoi s'intéresser à l'éthologie?

- Première raison : elle décrit et explique la formation de comportements collectifs intentionnels, qui ne sont pourtant pas menés par des individus intelligents.
  - Principal "philosophe" : Daniel Dennett
- Deuxième raison : Elle définit des modèles comportementaux individuels plus simples que ceux proposés en psychologie.
  - Principale inspiration : K. Lorenz et N. Tinbergen, McFarland

Application à l'I.A.D.

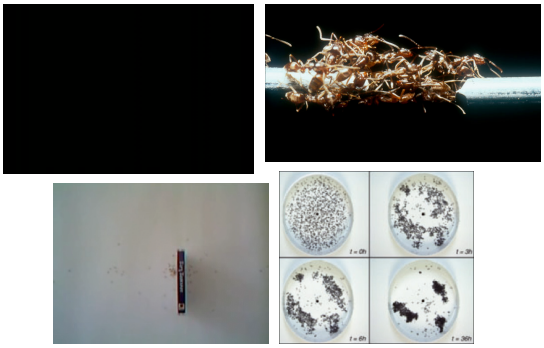
- L'éthologie offre des modèles d'organisation sociale ou de coopération souvent plus riches que la sociologie
  - intéressants pour la coopération d'agents hétérogènes
- Particularités de ces modèles
  - Les agents n'ont pas de représentation de l'organisation dont ils font partie.
  - Ils sont plus "simples" que les agents "sociaux" étudiés par la sociologie.
  - "L'intelligence" du système n'est plus une propriété des agents mais de la collectivité qu'ils forment.
- Synonymes
  - systèmes à fonctionnalités émergentes (ex: suivi de mur)
  - intelligence en essaim
  - systèmes auto-organisés

Principales métaphores

- Insectes sociaux, autres animaux sociaux
  - fourmis, termites, abeilles, guêpes
  - loups, rats, primates, oiseaux, poissons

- Exemple : les fonctions collectives des fourmis **Lasius Niger**
  - régulation de la température du nid (amplitude maximale : 1°C)
  - formation collective de ponts par les ouvrières
  - construction et protection des nids
  - tri du couvain et des items de nourriture
  - coopération dans le transport d'objets trop lourds
  - choix des chemins les plus courts entre nid et sources de nourriture
  - choix des sources de nourriture les plus riches au détriment des autres

Exemples de fonctions collectives chez les fourmis



Métaphores : transpositions des fonctions

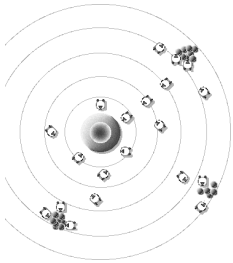
- C'est la voie la plus simple et la plus utilisée.
- Les éthologues construisent souvent eux-mêmes des modèles presque directement computationnels
  - robots fourrageurs
  - boids, etc.
- Mais le nombre de fonctions collectives directement transposables en fonctionnalités intéressantes (pour un exercice de conception informatique) est relativement réduit

Exemple 1 : le tri collectif

[Deneubourg & al. 1991][Drogoul 1993]

Exemple 2 : le fourragement Collectif

- "Benchmark" très populaire en IAD et Vie Artificielle
- Exploration et exploitation collective d'un environnement inconnu et dynamique, par des robots ou agents simulés
- Beaucoup de techniques envisageables (inspirées la plupart du temps des fourmis)



Métaphores : analogie des propriétés

- Plutôt qu'au résultat fonctionnel d'une collectivité, on s'intéresse à la dynamique collective ayant permis ce résultat, dans l'espoir de la généraliser à d'autres problèmes (artificiels)
- Approche plus générique et potentiellement plus riche, mais ... beaucoup plus compliquée (normal !)
  - compréhension des relations micro-macro
  - définition de modèles génériques de comportements
  - distribution des tâches
  - etc...

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

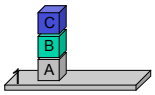
- Chaque agent a
  - un but = l'agent qui définit sa position finale
  - des accointances = les agents qu'il connaît
  - des gêneurs = les agents qui le gênent pour atteindre son but
  - des dépendances = les agents qui dépendent de lui pour pouvoir se satisfaire
- Le modèle de comportement est un automate 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)

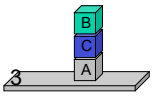
- Informellement...
  - Si le but d'un agent n'est pas satisfait, l'agent demande à son but de se satisfaire et se place dans ses dépendances.
  - Quand un agent se satisfait, il informe ses dépendances qu'elles peuvent se satisfaire.
  - Quand un agent ne peut se satisfaire, il recherche les gêneurs 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)

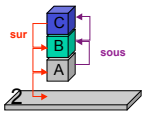
- Situation initiale



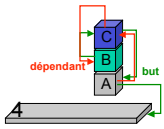
- Situation finale



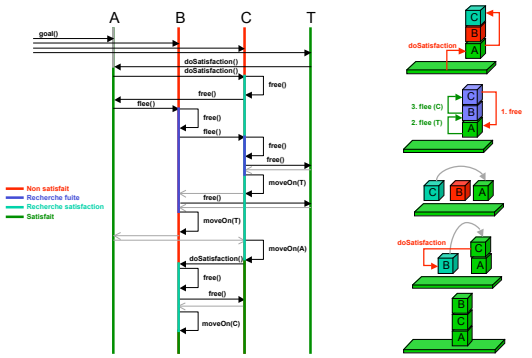
- Initialisation des accointances



- Initialisation des buts

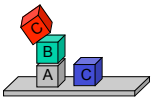


EPS: Exemple des cubes (2)



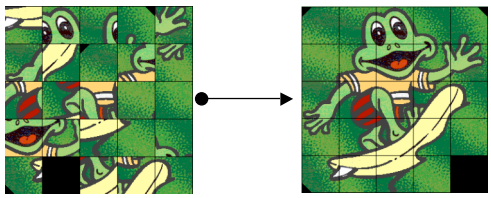
EPS: Exemple des cubes (3)

- 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*

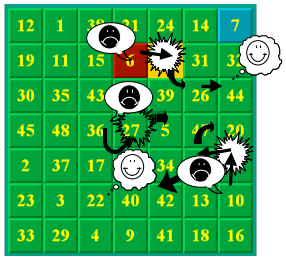
- parcours systématique de l'arbre avec attribution de scores aux différents états
- exploration prioritaire des états avec les meilleurs scores
- élimination des branches dont on sait avec certitude qu'elle ne peuvent pas donner de meilleur résultat qu'une solution déjà obtenue

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

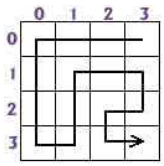
Taquin : Solution EPS (1)

- 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

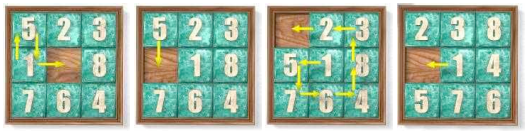
Taquin : Solution EPS (4)

- 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)

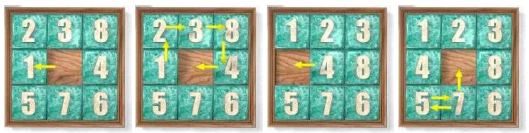
Taquin : Exemple 1.a



(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

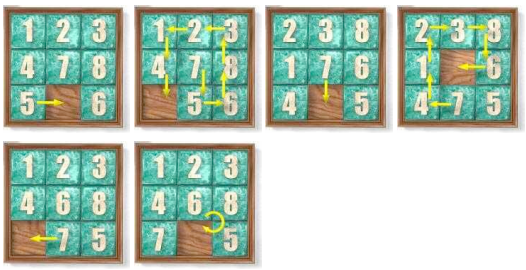
Taquin : Exemple 1.b



(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

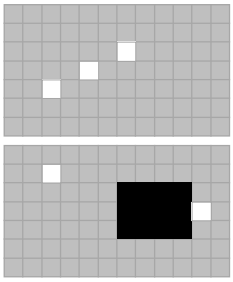


- Une fois 7 satisfait, la colonne 0 et la ligne 0 sont bloqués

Taquin : Exemple 2

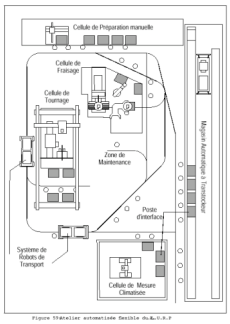
Taquin : résultats obtenus avec l'EPS

- 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.



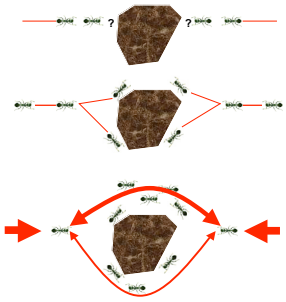
EPS : autres applis

- 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)



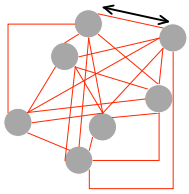
ANT – ACS: Présentation

- 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.



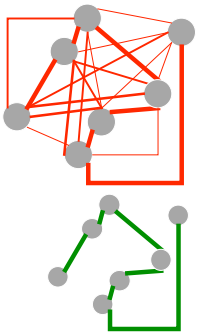
ANT: Voyageur de commerce (1)

- 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: Voyageur de Commerce (2)

- 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 :

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

où α et β 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
  - α et β, les valeurs contrôlant les poids respectifs des phéromones et de l'heuristique
  - ρ, 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)

Analogies des propriétés : problèmes

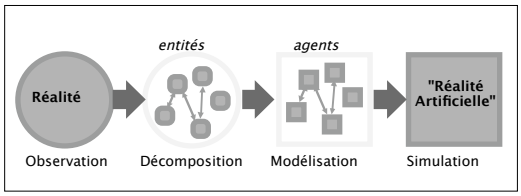
- Comprendre finement et être capable de reproduire le fonctionnement de systèmes auto-organisés réels
  - **Simulation multi-agent**
- Savoir comment distribuer une tâche collective parmi les agents en conservant les propriétés mentionnées
  - **Modèles d'agents**
  - **Méthodologie de conception**

Limite des métaphores: simulation

- Les métaphores sociologiques ou biologiques immédiatement applicables ne sont pas inépuisables.
- Il est difficile de les enrichir, car elles sont souvent intuitives (Contract Net, etc.).
- Développement d'une activité de simulation multi-agent à destination des sciences de l'homme et de la vie, afin de "découvrir" de nouveaux modèles.

Simulation multi-agent (1)

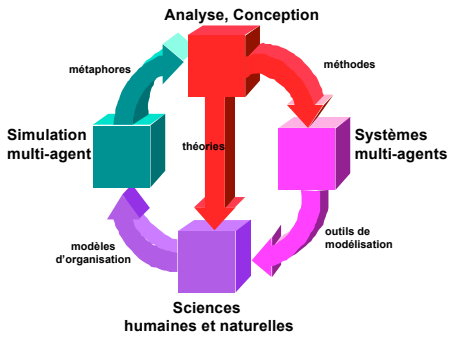
- Aussi appellée
  - IBM (Individual based modeling), ABS (Agent Based Simulation)
- Représentation et simulation de phénomènes complexes par la modélisation, sous forme d'agent, des entités qui les composent



Simulation multi-agent (2)

- Chaque agent est doté de capacités de comportements, d'interactions, de communication et de perception qui sont le reflet le plus fidèle possible de celles présentes au sein de l'entité dont il est le modèle informatique
- Cette technique de simulation permet de reproduire *in vitro* des phénomènes difficilement appréhendables de manière analytique
- Outil puissant de simulation, mais aussi méthode pour découvrir de nouvelles règles d'interaction sociale ou d'organisation

Le cycle "idéal" de la recherche en IAD



Agents réactifs : principales Caractéristiques

- Décomposition structurelle plutôt que fonctionnelle
  - grand ou très grand nombre d'agents
- Architectures minimales d'agents :
  - ex: stimulus/réponse
- Pas de représentations explicites (mémoire faible)
  - peu de mémoire individuelle
  - mémoire partagée grâce à l'environnement
- Communication via l'environnement
  - ex: propagation de signaux
- Organisation implicite/induite
  - auto-organisation
  - approche bottom-up
- Validation expérimentale

Propriétés des systèmes auto-organisés (1)

- Robustesse
  - redondance parmi les agents
  - tolérance accrue aux pannes individuelles
- Adaptativité
  - pas de contrôle central
  - pas de planification globale
  - adaptation locale et différentielle au contexte
- Réactivité
  - perceptions et actions distribuées et fortement couplées
  - pas de maintien d'états globaux (pas de backtrack)
- Simplicité
  - "économie" cognitive (cf. transp. suivant)
  - importante modularité

Propriétés des systèmes auto-organisés (2)

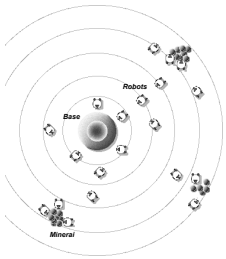
- "Simplicité" cognitive
  - pas de représentations symboliques
  - pas (ou peu) de mémoire individuelle
  - schémas de comportement basés sur des liaisons directes perception-action
  - modèles de comportement rigide (pas de méta-contrôle)
  - pas de communication intentionnelle (propagation, etc.)
  - utilisation importante de l'environnement (guide, mémoire partagée, médium de communication, etc.)
- Cette simplicité impose un fonctionnement collectif si l'on souhaite construire des systèmes intelligents

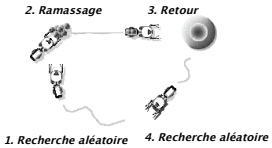
Démarche de conception

- Principe de parcimonie
  - Il n'est pas nécessaire de faire appel à des agents de granularité élevée pour résoudre un problème s'il s'avère que des agents dotés d'une granularité inférieure sont capables de trouver une solution
- Corollaire 1: principe de réductionnisme
  - Avant de faire appel à des agents d'une granularité donnée pour résoudre un problème, il convient de vérifier que des agents de granularité inférieure ne sont pas en mesure de lui donner une solution
- Corollaire 2: principe d'incrémentalité
  - On ne doit faire appel à des agents d'une certaine granularité que si les agents de granularité inférieure sont incapables de donner une solution au problème posé

Exemple de conception incrémentale : les Robots Fourrageurs

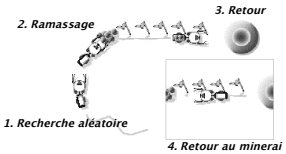
Pb = Collecte d'échantillons de minerai dans un environnement inconnu





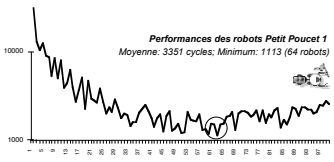
Les robots les plus simples

Premier ajout : Robots "Petit Poucet 1"

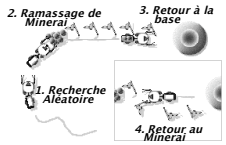


Tentative de coopération par partage d'informations et structuration de l'environnement

Les résultats sont trop variables d'une population à l'autre. L'information partagée peut ne pas refléter l'état réel de l'environnement

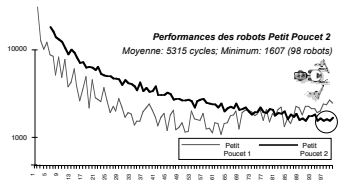


Deuxième ajout : Robots “Petit Poucet 2”

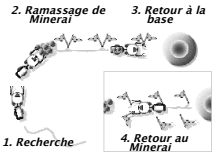


Le partage d'informations est plus dynamique et reflète l'état réel de la disponibilité en minerai

Mais la dynamicité est trop importante et les résultats sont dans l'ensemble moins bons que les précédents.

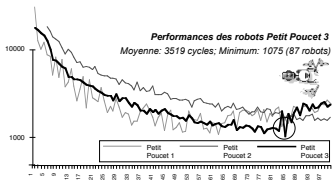


Troisième ajout : Robots “Petit Poucet 3”



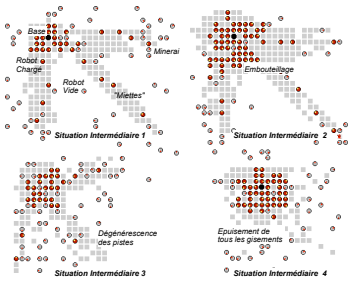
Dynamicité moins élevée du partage d'informations

Bonnes performances, sauf pour les grandes populations. La compétition inter-individuelle entraîne des phénomènes de blocage dus à une trop grande densité.

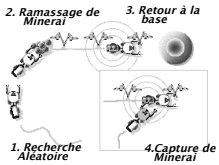


Robots “Petit Poucet 3”

Illustration des phénomènes de blocage

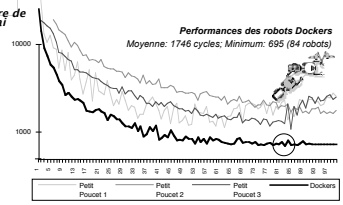


Quatrième ajout : les Robots “Dockers”



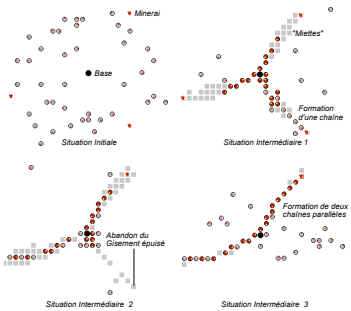
Ajout d'un comportement de "prise opportuniste"

Les résultats sont meilleurs que ceux des robots “Petit Poucet” et surtout stables à partir d'une certaine densité.



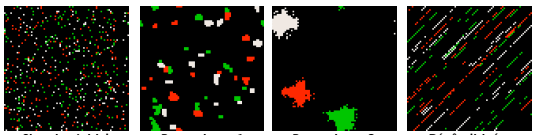
Les Robots “Dockers”

Emergence de chaînes, structures sociales qui permettent d'optimiser le comportement collectif (à partir de l'interaction de comportements individuels non coopératifs...)



Problèmes de conception (1)

- Espace des paramètres
  - Une modification minime d'un paramètre individuel peut totalement modifier la forme collective
  - Exemple : tri collectif



Situation Initiale | Perception = 1 (cases alentour) | Perception = 2 (cases alentour) | Dépôt dirigé (position + (1,1))

Problèmes de conception (2)

- Nombre et densité d'agents
  - Certains phénomènes ne surviennent qu'en présence d'un nombre ou densité d'agents donné : effet de "seuil" ou "masse critique"
- Exemples
  - construction des arches par les termites
  - ponts chez les fourmis
  - choix du plus court chemin




Problèmes de conception (3)

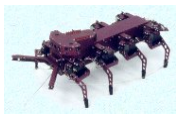
- Choisir un modèle de communication
  - propagation par l'environnement
    - champs de potentiel
    - communication non-intentionnelle
    - instantanée ou non
    - pérenne ou non
  - message point à point
  - message "broadcasté"

Problèmes de conception (4)

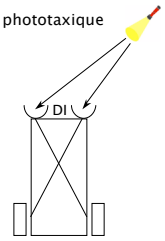
- Choisir un modèle de comportement
  - Pourquoi un comportement se déclenche
    - Perceptions ? Buts externes ?
    - Buts internes ? Motivations ?
    - Combinaison des deux ?
  - Comment un comportement se déclenche
    - Interactions entre perceptions et motivations (activation, planification)
    - Interactions entre comportements (sélection)
    - Inné ? Acquis ?

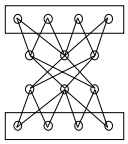
Architecture neuronale

Robot octopode



Robot phototaxique

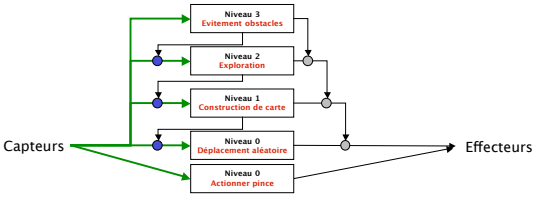




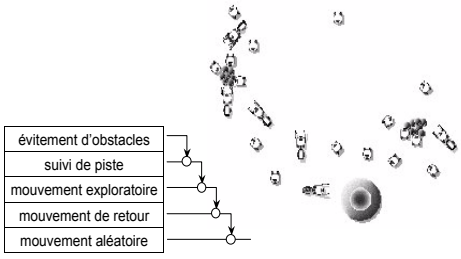
capteurs & actionneurs

Modèles : Architecture de Subsumption

- Brooks 86 – *Behavior Language* 90
- Modèle semi-hiérarchique de sélection d'actions

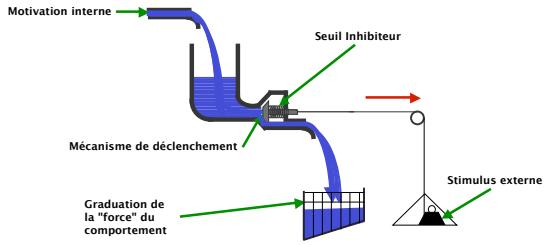


Ex: Robots fourrageurs [Steels]



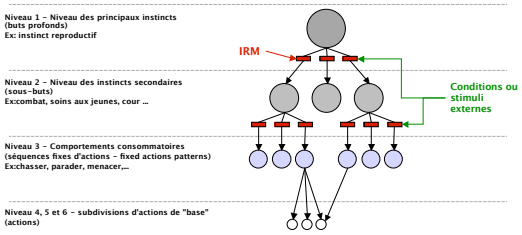
Modèles en éthologie: I.R.M.

- Konrad Lorenz (~1950, Innate Releasing Mechanism)
- Modèle d' activation de comportement



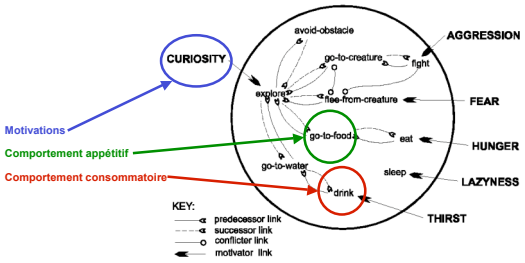
Modèles en éthologie: Tinbergen

- Modèle hiérarchique de sélection de comportements
- Distingue entre comportements appétitifs et consommatoires



Modèles : ANA (1)

- Patti Maes 91 – Agent Network Architecture
- Modèle semi-hiérarchique de sélection d'actions



Modèles: ANA (2)

- Caractéristiques:
  - Mixte Ethologie / Vie Artificielle
  - Bien adapté à des agents ayant des comportements différents
  - Forte redondance des actions de base (*flee-from-...*) en cas de comportements similaires
  - Difficile à concevoir et à tester
    - multiplicité des liens
    - dynamique importante
  - Bonne ouverture à l'apprentissage
    - renforcement des liens

Modèles : EMF (1)

[Drogoul 91 – 98]

- Modèle non hiérarchique de sélection de tâches (*IRM + fixed action patterns)*