UNIVERSITÉ LA ROCHELLE

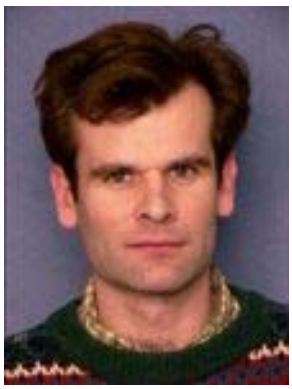
iup
Institut universitaire professionnalisé
Génie Informatique

# Rapport d'Intelligence Artificielle

## Licence IUP GI 2ème Année

## PRESENTATION DE

# L'EQUIPE



Dans l'ordre :

Baptiste MOUGEL, David CHANTOISEAU, Fabien NG FOK, Ronan LANGLO
Stéphane BAZEILLE, Yan BONNEL et Zacharie PICAUT.

Accompagné de leurs professeurs d' Intelligence Artificielle :
CHRISTOPHE DEMKO et PIERRE LOONIS

# Sommaire

# Introduction

L'expression Intelligence Artificielle s'est largement répandue dans le public, au fur et à mesure des progrès de la technologie informatique. Aujourd'hui l' Intelligence Artificielle fait partie de notre culture comme en témoigne l'existence de nombreux articles, livres, ou films s'y rapportant plus ou moins directement.

L'histoire de l'intelligence artificielle se divise en trois étapes principales qui, à bien des égards, sont devenues des axes de recherche simultanés puisque aucun axe n'a été définitivement abandonné au profit d'un autre. Ces axes définissent trois façons d'interpréter le rôle des mathématiques : comme calcul et comme symbole d'une part, comme permettant l'analyse des situations de l'autre.

L' intelligence artificielle, c'est réussir à donner à des machines des capacités leur permettant d'effectuer des tâches ou des activités réputées intelligentes. Son étude fournit de nouveaux repères, de nouveaux points de comparaison pour la compréhension de l'intelligence, et d'autre part elle s'inspire de ce que l'on sait du fonctionnement du cerveau et de la façon dont l'homme raisonne ( rien ne dit que l'intelligence Artificielle doive copier l'intelligence humaine dans toutes ses manières de procéder).

Les recherches en Intelligence Artificielle tendent à rendre la machine capable d'acquérir de l'information, de raisonner sur une situation statique ou dynamique, de résoudre des problèmes combinatoires, de faire un diagnostic, de proposer une décision, un plan d'action, d'expliquer et de communiquer les conclusions qu'elle obtient, de comprendre un texte ou un dialogue en langage naturel, de résumer, d'apprendre, de découvrir. Pour ce faire, la machine doit être munie de méthodes génériques susceptibles de s'adapter à de larges classes de situations.

Dans notre étude nous allons développer trois aspects bien distincts de l' Intelligence Artificielle : la résolution de problèmes , les stratégies de jeu , la reconnaissance des formes ; en comparant les différentes approches pour un même problème.

# PARTIE 1 : La résolution des problèmes

En intelligence artificielle, les problèmes sont résolus à l'aide d'algorithmes de recherche. Le principe général de ces algorithmes consiste d'abord à modéliser un état du problème. Un état du problème est codé en fonction de ces caractéristiques. L'objectif de l'algorithme est alors de trouver un état solution le plus rapidement possible.

Il existe deux principales familles d'algorithme qui proposent chacune une approche différente du problème.

Tout d'abord, nous avons étudié l'**approche déterministe** qui consiste à appliquer une stratégie de recherche définie au préalable afin de trouver une solution sans toutefois être sur d'y parvenir. Parmi les méthodes de résolution existantes, nous présenterons les suivantes :

- La méthode de recherche en profondeur
- La méthode de recherche en profondeur limitée
- La méthode de recherche en profondeur itérative
- La méthode de recherche en largeur
- La méthode du meilleur d'abord
- La méthode gloutonne
- La méthode en coût uniforme
- La méthode A*
- La méthode IDA*
- La méthode SMA*

Chacune de ces méthodes correspond à une stratégie de parcours des états possibles du problème sachant que l'ordinateur à une connaissance complète des coups possibles à un moment donné.

La seconde approche est l'**approche probabiliste ou stochastique**. Contrairement à la démarche précédente, il n'y a pas ici de méthode de recherche prédéfinie. On introduit une notion de qualité d'une solution (i.e. un état du problème) à partir de laquelle on oriente la poursuite de la recherche. En approche probabiliste, nous allons présenter :

- L'algorithme génétique

# I) L'approche déterministe :

## 1) Principe

On privilégie ici l'utilisation de fonctions génériques qui permettent de résoudre plusieurs types de problème en ne modifiant alors que les structures de données (en fonction de la méthode de résolution choisie) et les fonctions propres à la modélisation du problème.

Le principe d'un algorithme possible de résolution déterministe est le suivant :
L'algorithme est le même quelque soit le problème. Sa séquence d'exécution est la suivante :

1) Récupération de l'état courant (état à explorer)
2) Contrôle si l'état courant est solution
    a. Si oui : problème résolu
    b. Si non : étape suivante
3) Création des successeurs de cet état
4) Incorporation des successeurs à explorer
5) Retour à l'étape 1)

Les fonctions utilisées dans les étapes 1 et 4 varient en fonction de la méthode choisie. Pour chaque méthode la structure de donnée diffère.

Les fonctions utilisées dans les étapes 2 et 3 varient en fonction du problème à résoudre. Pour chaque problème la modélisation d'un état diffère.

## 2) Présentation des méthodes

Pour chacune des méthodes, le principe de la stratégie est brièvement expliqué. Puis on présente :

- les structures de données utilisées.
- les propriétés de la stratégie : sa complétude (si on est sur de trouver une solution), sa complexité en temps et en espace (temps d'exécution et espace mémoire utilisé) ainsi que son optimalité (si la solution trouvée est optimale).
- Un exemple de type de problème lequel cette stratégie est relativement adaptée.
- Une brève analyse des avantages et des inconvénients.

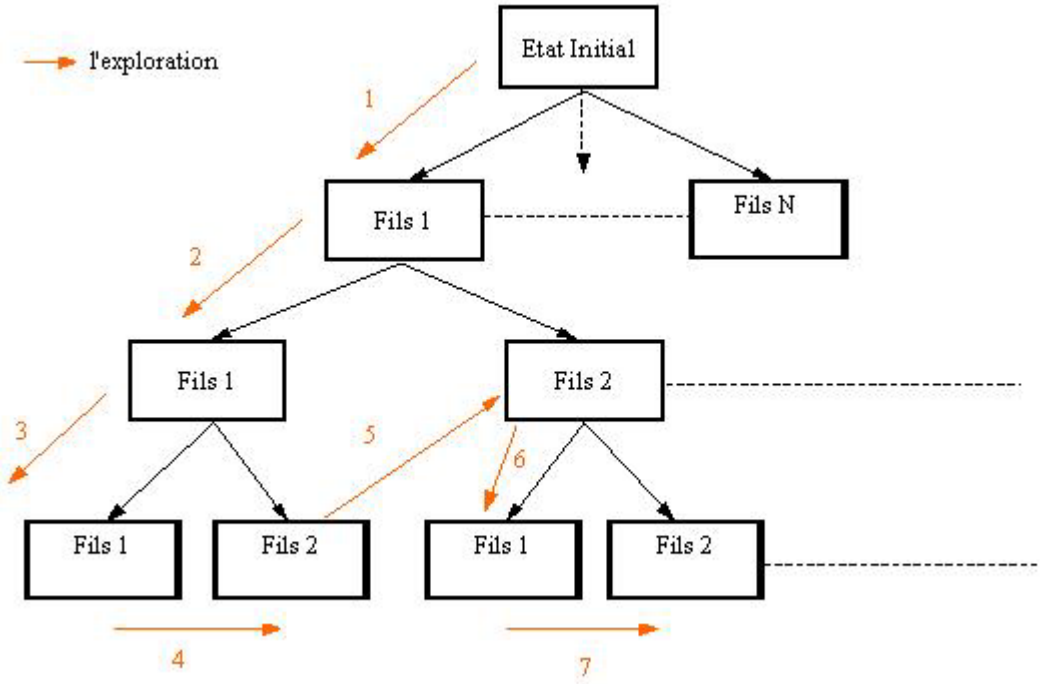
### Les méthodes de recherche aveugle :

Les méthodes suivantes sont des algorithmes de recherche aveugle. Ils n'exploitent aucune information concernant la structure de l'arbre de recherche ou la présence potentielle de nœuds solution pour optimiser la recherche.

## a) La méthode de recherche en profondeur

Description :

Si un état n'est pas solution, on explore son premier fils si il existe, sinon on explore son frère. Si ce dernier n'existe pas on explore son oncle. Ainsi, on explore d'abord un état dans chaque profondeur.




Structure utilisée :

Le principe de recherche est le suivant : la structure utilisée est une pile. La propriété d'une telle structure (Last In First Out) est adaptée à la logique de la méthode. Les fonctions d'extraction de l'état courant et d'incorporation des successeurs sont respectivement des fonctions de dépilement et d'empilement.

Propriétés :

- Complétude : non
- Complexité en temps : $o(b^n)$
- Complexité en espace : o(bm)
- Optimalité : non

Exemple d'utilisation :

Pour le problème des reines, on arrive rapidement à une solution plus susceptible d'être solution car on explore en profondeur d'abord.

Avantages :

Dans certains problèmes cette solution est adaptée. Lorsque l'on doit atteindre un certain niveau de profondeur pour avoir une solution par exemple.

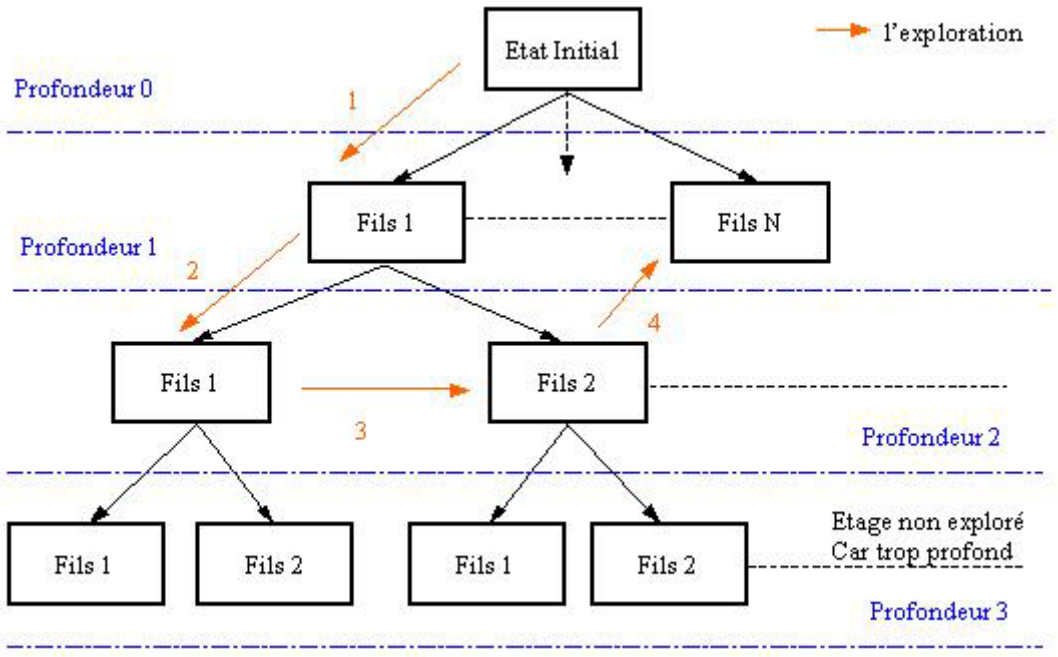
Inconvénients :

Cependant cette solution n'est pas optimale et elle est peu efficace. Dans certains cas ou un état est rencontré plusieurs fois, l'algorithme peut boucler sur la même séquence d'états, ce qui rend la recherche infinie et donc le problème insoluble.

## b) La méthode de recherche en profondeur limitée

Description :

Le principe est le même que pour la recherche en profondeur mais ici on fixe une profondeur limite. Lorsque celle ci est atteinte, l'exploration des états en profondeur cesse et on explore alors un état de même profondeur ou le cas échéant, de profondeur inférieure. Ainsi, on explore d'abord un état dans chaque profondeur jusqu'à la profondeur limite.




Ici, la profondeur limite est de 2.

Structure utilisée :

La structure utilisée est une pile. La propriété d'une telle structure (Last In First Out) est adaptée à la logique de la méthode. Les fonctions d'extraction de l'état courant et d'incorporation des successeurs sont respectivement des fonctions de dépilement et d'empilement. La fonction d'incorporation n'empile les états fils que si leur profondeur ne dépasse pas la limite.

Propriétés :

- Complétude : non
- Complexité en temps : $o(b^{min(d,p)})$ (d = profondeur de la solution, p = profondeur limite)
- Complexité en espace : o(bmin(d,p))

- Optimalité : non

Exemple d'utilisation :

Pour le problème du passeur, cette solution permet d'éviter de boucler sur une séquence d'état. Elle est adaptée notamment pour des problèmes dont les états sont récurrents.

Avantages :

Cette méthode évite de boucler.

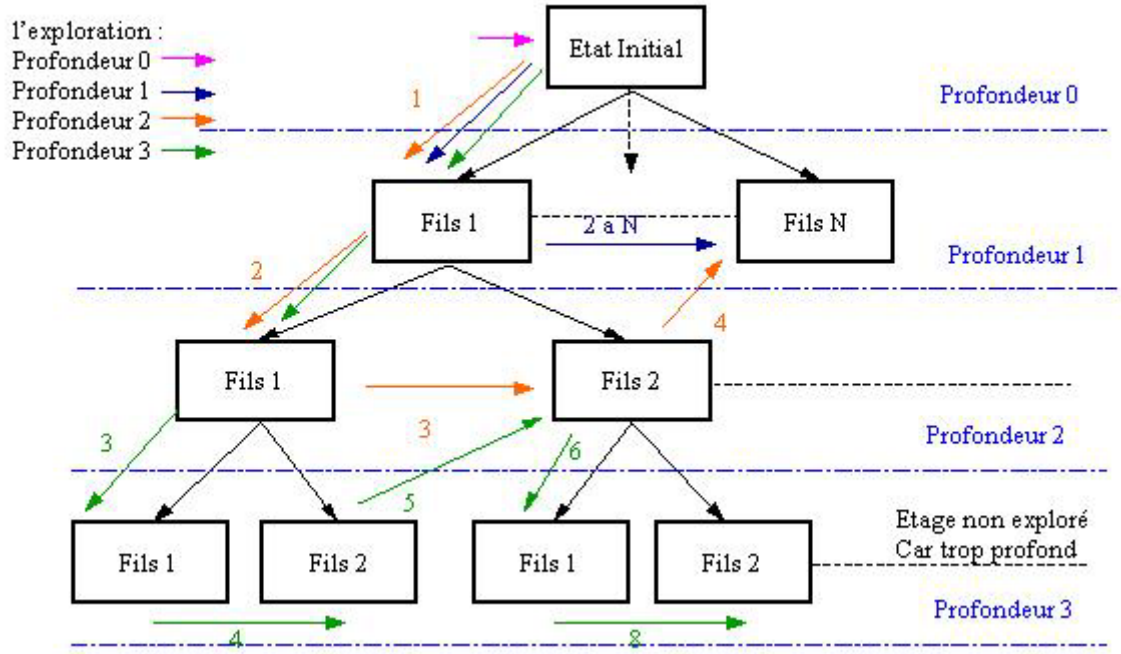
Inconvénients :

Hormis le fait que cette solution n'est pas optimale et ne présente pas de complétude, le choix de la limite reste un facteur important qui peut empêcher d'arriver jusqu'à la solution.

### c) La méthode de recherche en profondeur itérative

Description :

Le principe est le même que pour la recherche en profondeur limitée mais ici on fait varier la limite de façon à ce que les profondeurs soient explorées successivement en partant de la l'état initial. Ainsi, on effectue des recherches en profondeur limitée successives.




Structure utilisée :

La structure utilisée est une pile. La propriété d'une telle structure (Last In First Out) est adaptée à la logique de la méthode. Les fonctions d'extraction de l'état courant et d'incorporation des successeurs sont respectivement des fonctions de dépilement et d'empilement. La fonction d'incorporation n'empile les états fils que si leur profondeur ne dépasse pas la limite.

Propriétés :

- Complétude : oui
- Complexité en temps : $o(b^d)$ (d = profondeur de la solution, p = profondeur limite max)
- Complexité en espace : o(bmin(d,p))
- Optimalité : non

Exemple d'utilisation :

Pour le problème du passeur, cette solution permet d'éviter de boucler sur une séquence d'état. Elle est adaptée notamment pour des problèmes dont les états sont récurrents.

Avantages :

Cette méthode évite de boucler.

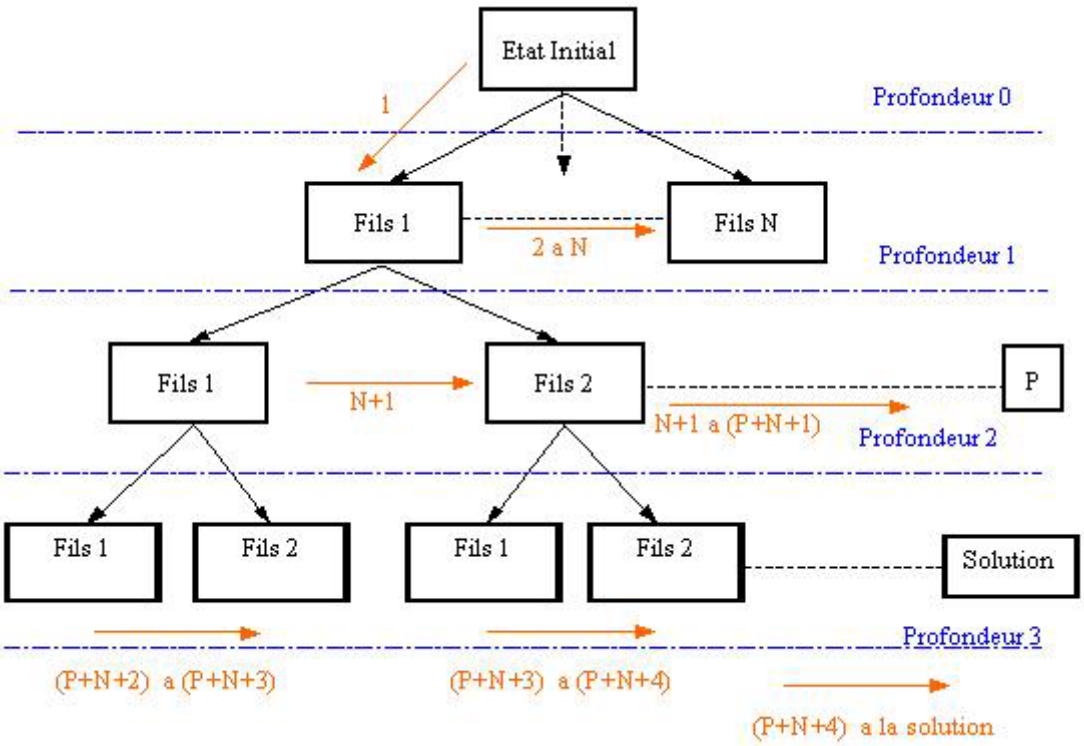
Inconvénients :

Hormis le fait que cette solution n'est pas optimale et ne présente pas de complétude, le choix de la limite reste un facteur important qui peut empêcher d'arriver jusqu'à la solution.

## d) La méthode de recherche en largeur

Description :

La recherche s'effectue d'abord sur la profondeur courante puis sur la profondeur suivante.



Structure utilisée :

La structure utilisée est une file. Cette structure est adaptée à la logique de recherche en largeur. Les fonctions d'extraction de l'état courant et d'incorporation des successeurs sont respectivement des fonctions d'enfilement et de défilement. Ainsi, on enfile tous les successeurs d'un état.

Propriétés :

- Complétude : oui
- Complexité en temps : $o(b^d)$
- Complexité en espace : $o(b^d)$
- Optimalité : oui

Exemple d'utilisation :

Pour le problème du passeur, cet algorithme permet une recherche efficace.

Inconvénients :

Cette méthode n'optimise pas l'utilisation de la mémoire puisque tous les fils d'une même profondeur sont empilés en même temps.

## Les méthodes heuristiques :

Les méthodes qui suivent utilisent des heuristiques ce qui améliore la performance. Un heuristique est une fonction qui associe à un état une mesure de désirabilité. Un algorithme de recherche heuristique utilise l'information disponible pour rendre le processus de recherche plus efficace.

### e) La méthode du meilleur d'abord (best-first search)

Description :

Se rapproche de la profondeur mais on développe d'abord le meilleur fils(celui qui a le la plus forte qualité estimée vis a vis du but.

Structure utilisée :

un tas : une pile organisée. Les nœuds y sont insérés dans l'ordre décroissant de leur qualité.

Propriétés :

- Complétude : non
- Complexité en temps : $o(b^n)$
- Complexité en espace : o(bn)
- Optimalité : non

Cas particuliers : la méthode gloutonne ou la méthode A*

Exemple : les problèmes de calcul de trajet routier.

Avantages : Solution trouvée sans avoir besoin de calculer tous les nœuds. De plus, on ne risque pas de boucle.

Inconvénients : pas de complétude

## f) La méthode gloutonne (greedy search)

Description :

La fonction d'évaluation utilisée est une heuristique : elle estime le coût du parcours entre n et le but développe le nœud qui semble le plus proche. La qualité de cette méthode est fonction de la qualité de l'heuristique utilisé.

Structure utilisée :

un tas : une pile organisée.

On explore le nœud qui semble le plus proche de la solution.

Propriétés :

- Complétude : non
- Complexité en temps : $o(b^n)$
- Complexité en espace : $o(b^n)$
- Optimalité : non

Exemple d'utilisation : calcul de plus courte distance.

Avantages : relativement efficace avec une bonne heuristique.

Inconvénients : Possibilité de boucler

## g) La méthode en coût uniforme

Description :

La fonction d'évaluation permet de calculer le coût du
Développer d'abord les nœuds de moindres coût.

Structure utilisée :

Un tas. Les successeurs sont inséré en fonction de leur coûts.

Propriétés :

- Complétude : oui
- Complexité en temps : $o(b^d)$
- Complexité en espace : $o(b^o)$
- Optimalité : oui

Exemple d'utilisation :

Avantages : optimal et complet

Inconvénients : peu efficace

## h) La méthode A*

Description :

Cette méthode évite d'explorer les parcours qui semblent les plus coûteux L'heuristique utilisée prend en compte deux facteurs : le coût du parcours pour atteindre un état et l'estimation du coût du parcours de cet état à la solution. C'est donc une combinaison du principe de la recherche gloutonne et de celle du coût uniforme.

Structure utilisée :

un tas : une pile organisée.

Propriétés :

- Complétude : oui (sauf s'il y a un nombre infini de nœuds avec une valeur de $f <= f(G)$)
- Complexité en temps : $o(b^d)$
- Complexité en espace : $o(b^d)$
- Optimalité : oui (avec heuristique admissible)
-

Exemple d'utilisation :

Calcul de trajet sous Age of Empire.

© Ensemble Studios – La recherche de chemin de Age Of Empires est gérée par un algorithme de type A*.

## i) La méthode IDA*

Description :

IDS effectue l'approfondissement par rapport à la profondeur limite de recherche ($L$ = 0, 1, 2, 3, ….). chaque itération de IDA* est une recherche en profondeur qui étend tous les nœuds à l'intérieur d'un contour délimité par la valeur courante de *f limite* en regardant par dessus cette limite pour savoir où se trouvera la prochaine valeur de *f limite.*

Structure utilisée : la même que A*

Propriétés :

- Complétude : oui
- Complexité en temps : $o(b^d)$
- Complexité en espace : $o(b^d)$
- Optimalité : oui

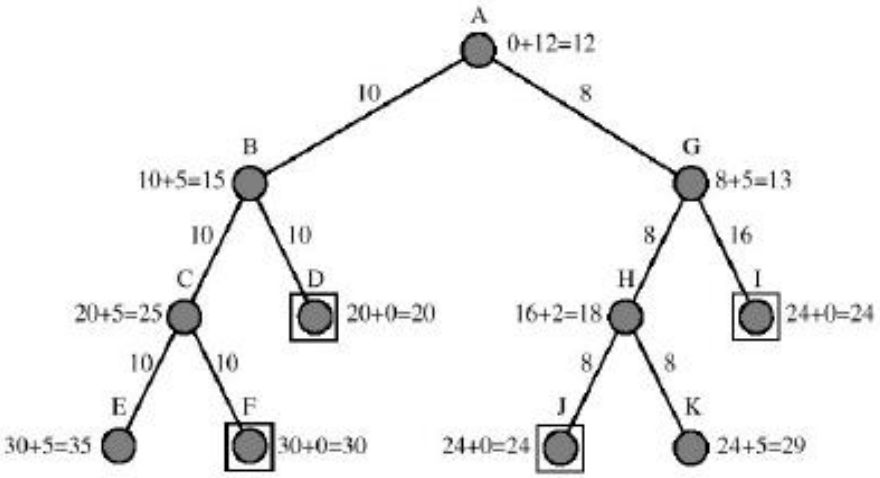
## j) La méthode SMA*

Description :

SMA* effectue sa propre gestion de la mémoire. Le principe est le suivant :

- si la mémoire (file d'attente) est pleine, alors faire de la place en éliminant le nœud le moins intéressant (celui avec une valeur *f* élevée) – retenir dans le nœud-ancêtre la valeur du meilleur descendant oublié

- dans tous les cas il retourne la meilleure solution accessible étant donné l'espace mémoire alloué




- Espace de recherche
  - chaque nœud est étiqueté avec sa valeur de f=g+h
  - les nœuds-solution sont D, F, I, J
- Objectif: trouver le nœud-solution de moindre coût avec un espace-mémoire suffisant pour contenir au maximum 3 noeuds

Structure utilisée : la même que A*

Propriétés :

- Complétude : oui, si l'espace mémoire alloué est suffisant pour contenir le chemin solution le moins profond
- Complexité en temps : $o(b^d)$
- Complexité en espace : $o(b^d)$
- Optimalité : oui, si l'espace mémoire alloué est suffisant pour contenir le chemin solution optimal le moins profond

Exemple d'utilisation :

Avantages :

- il évolue dans un espace mémoire alloué par avance
- il évite la duplicités des états
- il est complet

Inconvénients :

# 3) Tableaux récapitulatif

| Nom de la méthode | Complétude | Optimalité | Complexité en temps | Complexité en espace |
|---|---|---|---|---|
| Recherche en profondeur | Non | Non | $o(b^n)$ | o(bm) |
| Recherche en profondeur limitée | Non | Non | $o(b^{min(d,p)})$ | o(bmin(d,p)) |
| Recherche en profondeur itérative | Oui | Non | $o(b^{min(d,p)})$ | o(bmin(d,p)) |
| Recherche en largeur | Oui | Oui | $o(b^d)$ | $o(b^d)$ |
| Le meilleur d'abord | Non | Non | $o(b^n)$ | o(bm) |
| Gloutonne | Non | Non | $o(b^n)$ | o(bm) |
| Coût uniforme | Oui | Oui | $o(b^d)$ | $o(b^o)$ |
| A* | Oui | Oui | $o(b^d)$ | $o(b^d)$ |
| IDA* | Oui | Oui | $o(b^d)$ | $o(b^d)$ |
| SMA* | Oui | Oui | $o(b^d)$ | $o(b^d)$ |

# II) L'approche Probabiliste :

On utilise des lois de probabilité afin d'énumérer les solutions et de réduire le champ de recherche.

## Algorithmes génétiques

Lorsque les méthodes de recherches vues précédemment échouent ou ne sont pas adaptées, on utilise les algorithmes génétiques.

La programmation génétique est une extension des algorithmes génétiques aux programmes. Les algorithmes génétiques sont des algorithmes d'exploration fondés sur la sélection naturelle et la génétique. Ils utilisent les principes de la survie des structures les mieux adaptées, les échanges d'informations pseudo aléatoires.

Le principe des algorithmes génétiques s'inspire des comportements biologique des populations naturelle (théorie de DARWIN). On prend en compte une population de solutions au problème traité. On guide alors l'évolution par une fonction d'évaluation qui est maximisée au cours du processus.

On utilise un codage sous forme de gènes. Un génotype est associé à un problème et les gènes expriment les caractères de la solution codée. On reprend le principe de Darwin en utilisant des processus de recombinaison et de mutation qui permettent d'optimiser la recherche.Les propriétés des algorithmes génétiques sont :

- Les AG utilisent un codage des paramètres et non les paramètres eux-mêmes.
- Les AG travaillent sur une population de points et non sur un point particulier.
- Les AG n'utilisent que les fonctions étudiées, pas leur propriété (telle que la dérivabilité ou autre).
- Les AG utilisent des règles de transitions probabilistes et non déterministes.

Des exemple d'utilisation ou les algorithmes génétiques ont étés appliqués avec succès pour : la recherche de solution dans un jeu de stratégie, l'optimisation de réseaux de neurones, l'optimisation de fonctions (travaux de De Jong), la simulation de cellules biologiques, la reconnaissance de formes.

La difficulté majeure réside dans le problème du codage des individus et aussi dans le choix de la fonction d'évaluation. De plus les problèmes suivants peuvent se présenter :

- Une convergence prématurée : Bien que l'un des points forts des AG soit leur aptitude à s'échapper des minimas locaux, il peut arriver qu'ils convergent vers des solutions sous optimales. Cela arrive en particulier lorsque la population est trop petite, dans ce cas une solution peut apparaître rapidement et dominer alors que l'espace n'a pas été suffisamment exploré. Une méthode couramment utilisée est le ré échelonnage de la fonction d'évaluation, une augmentation de la probabilité de mutation (en fait ce taux peut varier au cours du temps conjointement à une variation inverse du taux de croisement).

- Un temps d'exécution trop important : cet inconvénient peut être lié à une population trop importante ou à une fonction d'évaluation trop coûteuse. Une solution possible pour le premier problème est de diminuer la population tout en augmentant le nombre de génération - dans ce cas, il faudra intégrer le risque d'une convergence prématurée. Pour le second problème, on pourra chercher à simplifier la fonction d'évaluation - ce faisant on perdra en précision. L'autre alternative étant de combiner évaluation simplifiée et évaluation complexe au cours du temps.

# III) En Résumé…

Toutes les méthodes vues sont plus ou moins efficaces en fonction du problème à résoudre. Dans le cas d'un problème plus simple avec un ensemble de solutions réduites, les algorithmes aveugles sont bien adaptés. Pour des problèmes plus complexes, les temps de calculs deviendraient trop long ou l'espace nécessaire devient trop important donc la recherche serait fastidieuse. Dans ce cas ci les algorithmes génétiques sont alors plus adaptés. Cette différence a pu être constaté lors des séances de TP notamment lors du traitement du problème des reines pour lequel les fonctions aveugles permettait de trouver une solution lorsque le damier était de taille réduite. Si la taille du damier devenait trop importante (> 20-25), les algorithmes génétiques s'imposaient.

De plus on peut constater que les méthodes de recherche en profondeur et en largeur sont complémentaires car l'une permet de trouver la solution à coup sûr malgré que le coup d'utilisation est élevé. Alors que l'autre a un coup de recherche relativement faible mais ne trouve pas la solution a coup sûr et il peut même arriver quel boucle si le problème n'est pas modélisé.

On peut, encore remarquer que la méthode de recherche en profondeur itérative à pour avantage de cumuler les avantages des recherches en largueur et profondeur. Mais le problème est que si la solution est a une profondeur importante, elle perd ses avantages car on doit reparcourir tout l'arbre a chaque itération.

Enfin, nous dirons que la liste de méthodes de résolution de problèmes décrites plus haut n'est pas exhaustive. En effet sur la toile d'araignée on trouve beaucoup d'autres méthodes dont voici quelques exemples :

- Hill Climbing (ou descente de gradient) :

  qui est une méthode équivalente à gravir l'Everest dans un épais brouillard et frappé d'amnésie. Ici, on se déplace constamment dans la direction de la valeur de pente la plus forte (croissante ou décroissante), mais aucun arbre de recherche n'est généré et si il y a plus d'un état-successeur possible, on choisit au hasard.

- Recuit ou Recuit simulé

  qui est une alternative au "hill climbing" aléatoire pour échapper à un maximum local *en réduisant graduellement la taille et la fréquence.*

  • recuit = processus physique de chauffage suivi d'un lent refroidissement pour obtenir une structure cristalline plus solide.

  • recuit simulé = processus qui baisse lentement sa "température" jusqu'à ce que le système se fige et que plus aucun changement ne soit observé.

# PARTIE 2 : Les Algorithmes de jeux ...

## - Sommaire -



## I) Pourquoi étudier les jeux ?

C'est le domaine favori de l'IA
– Les tâche sont bien structurées
– C'est à chaque fois un défi intellectuel
– Abstraction
– Mesure de performance
– et enfin l'IA ne nécessite pas obligatoirement de grandes quantités de connaissances.

On remarque cependant que tous les jeux ne sont pas adaptés à une étude par l'IA.
Nous avons privilégié dans ce rapport l'étude des jeux parfaits.

## II) Les jeux parfaits :

### 1) Les caractéristiques d'un problème de jeu

Présence d'un adversaire
– introduit un élément d'incertitude
– tous les mouvements ne sont pas contrôlés par l'ordinateur
• Les programmes de jeu doivent faire face à l'imprévu
• Complexité: les jeux intéressants sont trop complexes pour envisager une solution exhaustive
– exemple: les échecs ont un facteur de branchement de 35!
• Le but à chaque pas est de choisir le meilleur coup à jouer
• Optimalité: les coups doivent être évalués selon leur coût

## 2) Jeu et algorithmes de recherche

Jouer c'est rechercher le mouvement permettant de gagner
• Les jeux de plateau contiennent les notions de:
– état initial et état gagnant (but)
– opérateurs de transition (règles du jeu, déplacements de pièces)
• Une fonction heuristique permet d'estimer s'il y a gain, perte ou match nul
• Premiers algorithmes de jeu
– algorithme pour jeu parfait, J. Von Neumann (1944)
– horizon fini, évaluation/approximation, K. Zuse (1945), C. Shannon (1950),A. Samuel (1952)
– réduction de coût par élagage, McCarthy (1956)
• Mais il y a d'importantes différences avec un problème standard de recherche.

## 3) Les difficultés

• Les mouvements de l'adversaire ne sont pas toujours prévisibles.
– il faut donc être capable de prendre en compte toutes les situations.
• L'espace de recherche est généralement très vaste:
exemple: pour le jeu d'échecs :
– nombre de choix par coup: 35 (facteur de branchement)
– nombre moyen de coups par jeu: 50 (par joueur)
– nombre total d'états: $35^{100}$
– nombre de nœuds de l'arbre: $10^{40}$
• Il est donc impossible d'explorer tout l'espace de recherche pour trouver le meilleur mouvement à effectuer.

## 4) Les types de jeux

Il existe 3 grands types de jeu :
- les jeux déterministes avec information parfaite: échecs, jeu de dames, go, othello ….
- les jeux avec hasard et information parfaite : backgammon, monopoly …
- les jeux avec hasard et information imparfaite : bridge, poker, scrabble …

## 5) Éléments d'un système informatique de jeu

Composants de base d'un "moteur" de jeu:
• Générateur de mouvements
– génère les mouvements valides à partir de l'état courant
• Test de terminaison
– décide si l'état courant est un gain, une perte, un nul ou rien de particulier
• Fonction d'évaluation (fonction de gain ou d'utilité)
– évalue la qualité d'un état donné indépendamment des coups passés ou futurs
• Stratégie de contrôle
– fournit les éléments nécessaires pour choisir entre différentes options celle qui semble la plus prometteuse.

# III) Mini Max :

• Max et Min sont les 2 joueurs. Un gain pour Max est une perte pour Min et vice versa.

• +1 pour un gain, -1 pour une perte, 0 pour un nul.

• La somme des valeurs de la fonction d'évaluation pour les 2 joueurs est nulle à la fin du jeu.

• Il faut disposer d'une fonction d'évaluation statique capable de mesurer la qualité d'une configuration par rapport à un joueur (généralement Max)

– car il n'est pas possible de produire tout l'arbre de recherche jusqu'à la fin du jeu, c'est à dire au moment où la décision gain / perte / nul est claire.

## 1) Le Principe de Mini Max:

maximiser la valeur d'utilité pour Max avec l'hypothèse que Min joue parfaitement pour la minimiser,

– étendre l'arbre de jeu

– calculer la valeur de la fonction de gain pour chaque nœud terminal

– propager ces valeurs aux nœuds non-terminaux

- la valeur minimum (adversaire) aux nœuds MIN.
- la valeur maximum (joueur) aux nœuds MAX.

***Remarque :*** Le calcul est effectué "en profondeur" par l'utilisation d'une pile des appels récursifs.

## 2) Propriétés de Mini Max

Complet Oui si l'arbre de jeu est fini.

• Optimal Oui si l'adversaire est aussi optimal.

• Complexité en temps $O(b^m)$ où $b$ est le facteur de branchement et $m$ est l'horizon de recherche.

• Complexité en place $O(b\ m)$ (car exploration en profondeur) pour les échecs:
avec b = 35 et m = 100

L'algorithme *Mini Max* est totalement impraticable !

C'est pourquoi il existe des méthodes de réduction d'espace.

## 3) Méthodes de réduction d'espace

• Même pour le jeu de Tic – Tac - To la taille de l'espace de recherche est grande:
$3^9$ = 19'683 nœuds pour un damier entièrement occupé !

• Pour des jeux non-triviaux les coups doivent pouvoir être déterminés sans pour autant devoir générer tout l'arbre de jeu.

• Il faut alors pouvoir évaluer des nœuds non-terminaux

• Il existe ainsi 2 solutions:

– *Mini Max* avec profondeur limitée

– et Elagage (aussi appelé élagage alpha – bêta).

### 4) Mini Max avec profondeur limitée

• Principe

– étendre l'arbre de jeu jusqu'à une profondeur *N* à partir du nœud courant.

– calculer la valeur de la fonction d'évaluation pour chaque nœud-feuille (pas nécessairement terminal).

– propager ces valeurs aux nœuds non-terminaux jusqu'à la racine.

• pour être efficace, il est important d'avoir une bonne fonction heuristique d'évaluation !

– généralement on prend une fonction linéaire pondérant la valeur de chaque pièce du jeu ,du type $e = w_1 f_1 + w_2 f_2 + \ldots + w_n f_n$.

### 5) Algorithme Mini Max avec limite de profondeur

1. Étendre l'arbre de jeu à partir de l'état courant (où c'est à Max de jouer) jusqu'à une profondeur N.

2. Calculer la fonction d'évaluation pour chacune des feuilles terminales de l'arbre.

3. Rétro -propager les valeurs des nœuds - feuilles vers la racine de l'arbre de la manière suivante:

1. Un nœud Max reçoit la valeur maximum de l'évaluation de ses successeurs.

2. Un nœud Min reçoit la valeur minimum de l'évaluation de ses successeurs.

4. Choisir le mouvement vers le nœud Min qui possède la valeur rétro-propagée la plus élevée.

### 6) Mini Max avec profondeur limitée

• Le jeu d'échecs

si $b^m = 10^6$ et b = 35 m = 4 niveaux de prévision.

– prévision de 4 niveaux: joueur novice.

– prévision de 8 niveaux: niveau "maître" et bon programme PC.

- prévision de 12 niveaux: G. Kasparov et Deep-Blue.

## IV) Mini Max avec élagage :

Principe de l'élagage.

•étendre l'arbre de jeu jusqu'à une profondeur N par recherche en profondeur.

• ne plus générer les successeurs d'un nœud dès qu'il est évident que ce nœud ne sera pas choisi (compte tenu des nœuds déjà examinés).

### 1) propriétés de l'Élagage :

L'élagage n'affecte pas le résultat final.

• l'efficacité de l'élagage est fonction de l'ordre d'apparition des successeurs, complexité en temps.

• meilleur des cas: $O(b^{m/2})$

– permet de doubler la profondeur de recherche pour atteindre une prédiction sur 8 niveaux et ainsi jouer "dignement" aux échecs.

• pire des cas: identique à Mini Max.

• cas moyen: $O((b/\log(b))^m)$ [Knuth&Moore 75].

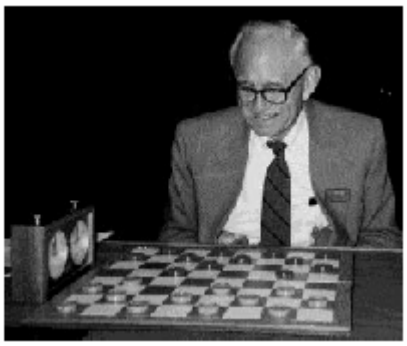
# V) L'évolution des différents Jeux :

## 1) jeu de dames :

• 1952: Samuel développe le 1er programme qui apprend sa propre fonction d'évaluation au cours du temps.

• 1992: Chinook (J. Schaeffer) gagne le championnat de USA. Ce programme utilise l'algorithme d'élagage.

• 1994: Chinook met fin à 40 ans de règne du champion du monde Marion Tinsley. Il utilisait une base de données de fin de parties définissant le jeu parfait pour toutes positions incluant 8 pièces ou moins, soit au total 443'748'401'247 positions.



Nom: Marion Tinsley
Profession: professeur de mathématiques
Hobby: le jeu de dames
Record: en plus de 42 ans n'a perdu que 3 (!) parties

## 2) Le jeu de backgammon :

• Présence du hasard (jets de dés) rend la recherche coûteuse.

• 1980: le programme BKG effectuant une recherche à un niveau et avec beaucoup de chance a battu le champion du monde.

• 1992: Tesauro, en appliquant les méthodes d'apprentissage de Samuel avec des réseaux de neurones artificiels, développe une nouvelle fonction d'évaluation et son programme est dès lors parmi les 3 meilleurs joueurs du monde.

## 3) Le jeu Othello

• les champions humains refusent de se mesurer à l'ordinateur qui est définitivement meilleur.

Nom: Takeshi Murakami
Titre: dernier homme champion du monde au jeu Othello

## 4) Le jeu d'échecs

• Jeu qui a reçu la plus grande attention.
• au début les progrès on été très lents.
• 1970: 1er programme à gagner le championnat ACM d'échecs.
informatiques, utilisant une recherche ,des ouvertures classiques et des algorithmes de fins de parties infaillibles.
• 1982: "Belle" est le premier ordinateur spécialisé dans le jeu d'échecs capable d'explorer quelques millions de combinaisons par coups.
• 1985: "Hitech" se classe parmi les 800 meilleurs joueurs du monde, il est capable d'explorer plus de 10 millions de combinaisons par coups.
• 1997: *"Deep Blu*e" bat G. Kasparov. Il effectue une recherche sur 14 niveaux et explore plus de 1 milliard de combinaisons par coup à raison de plus de 200 millions de positions par seconde, utilise une fonction d'évaluation très sophistiquée et des méthodes non divulguées pour étendre certaines recherches à plus de 40 niveaux.

***Kasparov VS Deep Blue.***

| | |
|---|---|
| **Nom** ***Kasparov*** | **Nom** ***Deep Blue*** |
| **Taille** 1 m83 | **Taille** 2 m20 |
| **Poids** 77 kg | **Poids** 900 kg |
| **Age** 34 ans | **Age** 4 ans |
| **"Ordinateur"**50 milliards neurones | **"Ordinateur"**512 processeurs |
| **Vitesse**2 pos/sec | **Vitesse**2 200,000,000 pos/sec |
| **Connaissance** Extensive | **Connaissance** Primitive |
| **Source d'énergie** Electrique/chimique | **Source d'énergie** Electrique |
| **Ego** Démesuré | **Ego** Aucun |

Garry Kasparov and Deep Blue.
© 1997 courtesy IBM.

# VI) L' exemple du jeu Puissance 4 pour la fonction heuristique

On se basera sur un algorithme de Mini Max…

Il faudra donc savoir attribuer une note à chacune des configurations terminales simulées au cours de l'exécution du Mini Max. La fonction d'évaluation est une des plus importante d'un jeu au niveau de sa pertinence. Nous attribuerons des poids à des alignements ou parties d'alignement, selon leur hauteur, leur potentiel de victoire…

Ainsi, un alignement de 4 (pions d'une même couleur) a un poids très important. Un alignement de 3 vaut moins cher, un alignement de 2 encore moins etc. De la même façon, un alignement haut vaut moins cher qu'un alignement bas…

Un alignement centré vaut plus cher qu'un alignement excentré…

Tenant compte de cet ensemble de considérations et appréciations, un coup futur, ami ou ennemi, pourra donc être évalué en première approche en calculant la somme des poids des alignements mono couleur, partiels ou non qui contiennent la case concernée par ce coup possible. La valeur de la fonction d'évaluation pour une configuration, mesurant son devenir, son potentiel, est alors en première approche la somme des valeurs des coups possibles.

On voit donc que le potentiel d'une configuration ou plateau de jeu peut être évalué en s'intéressant aux cases vides et aux alignements mono couleur possibles avec ces cases vides. La valeur d'une configuration repose donc sur ses cases vides et leur voisinage. On tiendra compte du fait qu'une case vide peut être "posée" sur la configuration en cours de simulation, ou au contraire, en "lévitation" par rapport à celle ci.

Bien entendu, dans le cas général, une case vide basse vaut plus cher qu'une case vide haute....

La machine a un signe positif, tandis que l'homme a lui un signe négatif.

Au passage, on peut remarquer que notre fonction est donc symétrique par rapport à 0…

On pourra donc optimiser notre Mini Max comme vu a l'examen. La valeur de la fonction d'évaluation ainsi conçue peut être calculée assez vite…

A ce stade, la fonction d'évaluation reste assez générale et peu spécifique au jeu du puissance 4…

Malgré cette simplicité l'ordinateur gagne la plupart du temps. Il gagne de trois manières :

-La première correspond à l'erreur « stupide » ou d'inattention de l'homme.

-La seconde est la victoire "au finish". Au finish, il n'y a pas victoire brillante ou coup monté, le plateau est quasi plein, une colonne ou deux sont « bloquées », une dernière colonne doit être montée et à un moment donné, la machine a la main pour réaliser un alignement au 36,.. 38, 39… 42 ième coup.

-La troisième est une victoire en stratégie pure, par exemple deux alignements simultanés, ou bien l'homme, obligé de jouer pour boucher des trous se heurte tout à coup à un alignement machine etc.

On remarque qu'à une profondeur assez importante l'homme n'arrive à gagner qu'au finish. On va donc maintenant s'attarder sur la stratégie pour la victoire au finish. Il est clair que certains alignements de 3 pions ont plus d'importance que d'autres.

Par exemple, si la machine a commencé la partie, un alignement machine de 3 pions avec pion manquant en ligne impaire est plus intéressant qu'un alignement de 3 avec pion manquant en ligne paire.

Cette constatation est cohérente avec le fait que si la machine a commencé, elle jouera son dernier pion au maximum en ligne 5.

Bien entendu, cette notion de pion manquant n'a de sens que si la case vide envisagée se trouve " en lévitation " par rapport à la configuration courante (ou plutôt à la configuration explorer par le Mini Max), puisque dans le cas contraire, elle est immédiatement accessible au joueur qui a la main.

Dans un cas de figure où il reste en finale une colonne à monter, on trouvera donc inéluctablement, sauf victoire anticipée, un pion homme en ligne 6, un pion machine en ligne 5, etc.

Ce raisonnement se transpose aisément au cas où l'homme a commencé la partie…
Dans la phase ultime d'une partie où chacun a tenu le coup, cette considération est donc très importante, très anticipative et donc correspondrai à une analyse à profondeur 30-40 et donc très intéressante.

Avec une telle fonction, on pourrait espérer que le système soit capable de tenir compte à la fois du long terme avec la notion très anticipative de pion manquant au bon niveau, et du court terme avec une incitation très tactique à construire les alignements les plus longs, les plus nombreux, les plus simultanés, les plus bas.

On a donc de cette manière une fonction d'évaluation qui exprime à la fois une stratégie à long terme en plaçant les pion de manière à avoir ses dernier coups qui tombent sur des alignements, et une stratégie plus immédiate qui cherche à produire le plus d'alignement possible le plus centré possible et le plus bas…

On obtient donc ainsi une fonction d'évaluation qui paraît très bonne, mais on voit avec ce problème qu'une fonction d'évaluation reste très spécifique au jeu étudier, il n'existe pas de fonction d'évaluation générale qui marcherait pour tous les jeux…
Il ne resterait plus qu'à mettre en place un réseau de neurones pour apprendre les meilleurs poids possibles.

# VII) L'INTELLIGENCE ARTIFICIELLE IMPOSSIBLE : LE JEU DE GO

## 1) Le jeu de Go

Il se joue à deux avec des pierres noires et blanches sur un damier de 19 lignes et 19 colonnes appelé le goban. Le but du jeu est de dominer plus d'intersections que son adversaire en connectant ses propres pierres et en capturant les pierres adverses.

## 2) Comparaison : Jeu de GO – Jeu d'échec

Le nombre de coups possibles dans une position typique avoisine en moyenne 300 au Go contre 40 aux Echecs.

Le nombre de coups joués dans une partie typique avoisine 250 au Go contre 100 aux Echecs.

Et enfin la taille très approximative de l'arbre des possibilités du jeu de Go est de $10^{600}$ alors que celui du jeu d' Echecs n'est « seulement » que de $10^{40}$ environ..

## 3) Le meilleur programme de Go actuellement a un niveau de débutant.

Depuis quelques années, plusieurs tournois d'ordinateurs sont organisés. En 1990, "Goliath", le programme hollandais de Mark Boon qui s'est imposé dans le tournoi des ordinateurs, a perdu une partie à 15 pierres de handicap contre un professionnel... Le niveau de "Goliath" est celui d'un joueur de club débutant.

## 4) La machine résout des sous-problèmes posés par le jeu de Go mais pas de problèmes général.

Jusqu'à présent les résultats obtenus de nombreuses études, même s'ils ouvrent des nouvelles possibilités, ne s'appliquent qu'à des sous-problèmes bien définis et limités du jeu de Go mais en aucun cas à sa globalité.

Pourquoi l'homme qui est inférieur à la machine quand il s'agit de combinatoire, est-il incomparablement plus fort qu'elle quand il s'agit de jouer au Go ?
Pour comprendre cela il faut faire des rapprochements avec monde réel..

## 5) Le jeu de Go et ses similitudes avec le monde réel.

L'espace réel dans lequel nous vivons est constitué d'une infinité de points dans trois dimensions. En chaque point, nous percevons diverses composantes (des couleurs, des odeurs, des sons, etc...) qui varient dans l'infinité du temps et nous y effectuons de multiples actions. Le goban sur lequel le joueur de Go joue ne possède qu'un nombre fini d'intersections sur deux dimensions. Chaque intersection est blanche, noire ou vide et varie un nombre fini de fois au cours de la partie. Les deux seules actions possibles sont la pose d'une pierre et la capture d'une chaîne de pierres. Mais si c'était la seule explication, l'homme serais également plus fort à l' othello, ce qui n'est pas le cas du tout comme nous avons vu. Le jeu de Go est un jeu d'occupation de l'espace par deux joueurs, Blanc et Noir qui luttent à la vie et à la mort, de la même manière que les espèces vivantes combattent pour préserver leur espace vital dans le monde réel. L'être humain s'est habitué à faire face aux situations rencontrées dans le monde réel. Pendant des dizaines de milliers d'années, il y a acquis des capacités de perception, de décision, d'adaptation et de communication grâce au langage pour y vivre. Le joueur de Go (qui a l'avantage d'être un humain!) profite de la richesse de ses capacités et les applique dans les parties de Go, bien piètres simplifications du monde réel.

## 6) Le jeu de GO est un jeu visuel.

Le Go est un jeu essentiellement visuel. L'œil d'un joueur de Go se balade sans cesse sur le goban en quête d'informations. Du premier coup d'œil, un joueur perçoit des formes de pierres, des surfaces, des territoires, des frontières, des zones d'influence, des "bonnes et des mauvaises formes", des équilibres de pierres, l'harmonie globale du jeu, des zones "patatoïdes" etc... Ce travail quasiment inconscient permet au joueur de Go de faire des regroupements d'intersections en classes d'équivalence et donc de réduire le nombre de coups envisagés.

La rapidité de perception de tous ces concepts est sans doute l'atout principal du joueur de Go humain par rapport à la machine. De nombreux algorithmes ont été proposés pour essayer de calculer les zones d'influence et les territoires potentiels induits par des configurations de pierres, mais il faut reconnaître que jusqu'à présent les résultats sont bien en dessous de ce que l'humain un peu exercé perçoit du premier coup d'œil.

## 7) La programmation du jeu de Go est un défi pour l'intelligence artificielle

Parce que jouer au Go requiert des facultés de perception, de décision, d'adaptation et d'utilisation de langage, le joueur humain qui vit dans le monde réel, possède un avantage inestimable sur la machine, incapable actuellement d'utiliser son avantage calculatoire dans ce jeu. La programmation du jeu de Go passe donc par une analyse du fonctionnement cognitif de l'homme et du joueur de Go. On pourra alors représenter les connaissances humaines de façon à les utiliser efficacement grâce aux possibilités calculatoires des machines. Ce travail s'inscrit dans le cadre de la recherche en intelligence artificielle qui trouve dans la programmation du jeu de Go un terrain de recherche idéal et un défi difficile.

Mais l'équipe qui arrivera à battre un joueur professionnel au jeu de GO remportera un prix de la bagatelle de 2.000.000 de dollars ! ! !
Donc bon courages à vous ;)

# VIII) CONCLUSION :

L'Intelligence artificielle est un domaine très vaste, et l'exemple des stratégies de jeux regroupe beaucoup de domaines de l'IA, allant des recherche A* pour les déplacement dans les jeux modernes, en passant par des recherche en profondeur type Mini Max pour les jeux classiques, mais également des réseaux de neurones dans les jeux où il doit apprendre en fonction de l'utilisateur, comme par exemple « Black and White » ou encore « Créature ».
Mais il reste des jeux où on est incapable de créer des IA correctes comme par exemple le jeux de GO ...

Ceci dit l'intelligence artificielle en plus d'être utilisé tous ces jeux dits « classiques » (échec ,dame , go …) est aussi utilisés dans beaucoup de jeux vidéos.

L'application principale de l'IA dans ces jeux est souvent liée au déplacement des acteurs, que ce soit les ennemis, ou le personnage du joueur .Certains jeux posent même plus de problèmes que d'autres, comme les jeux de stratégie, par exemple, qui doivent déplacer des groupes d'individus. Donner l'illusion de la personnalité et de l'intelligence à un acteur (tactiques, stratégies…) ou donner un comportement collectif à une troupe relève vraiment du domaine de l'IA.Exemple ci-dessous deux jeux utilisant l'intelligence artificielle.

Half Life et GTA 3

Avec le developpement de plus en plus pousse des jeux et l'interet grandissant que le public lui porte ,l'intelligence artificielle dans le secteur des jeux videos a encore aujourd'hui de beau jour devant elle.

# PARTIE 3 : Reconnaissance des formes

La notion de formes est à prendre au sens large. Les formes visuelles, sonores ou tactiles sont les différents types que l'on peut rencontrer.

Le domaine d'application de la reconnaissance des formes est la robotique et les interfaces homme/machine. L'idée de construire des machines capables de simuler des êtres humains afin de les aider dans certaines tâches, voire de les remplacer, était antérieure aux ordinateurs. Leur apparition a permis d'étendre le spectre des tâches à simuler en ajoutant celles dont l'exécution relève de facultés mentales comme la perception et le raisonnement.

Le problème que cherche à résoudre la reconnaissance des formes est d'associer une étiquette à une donnée qui peut se présenter sous forme d'une image ou d'un signal. Des données différentes peuvent recevoir la même étiquette, ces données sont les réalisations ou les exemplaires de la classe identifiée par l'étiquette. Par exemple, le son « a » prononcé par différents locuteurs conduit à des signaux différents mais ces différences ne sont pas significatives du point de vue de l'identification du son, ces signaux sont des réalisations de la classe « a ». De même, l'écriture manuscrite du caractère « a » varie d'un scripteur à l'autre mais le lecteur identifiera le caractère « a » pour chacune de ces réalisations.

Des méthodes générales ont été développées en reconnaissance des formes pour extraire automatiquement des informations des données sensibles afin de caractériser les classes de formes et d'assigner automatiquement des données à ces classes. La mise en œuvre de ces méthodes générales pour des problèmes particuliers amène à introduire la notion de processus de reconnaissance qui pose la question de l'intégration des méthodes de la reconnaissance de formes dans un système qui a pour but de reconnaître des formes.

Parmi les nombreux modèles proposés pour résoudre le problème de la reconnaissance de forme, les modèles neuromimétiques, ou réseaux neuronaux (Haton, 1995), occupent depuis environ dix ans une place notable. Ceci n'est pas pour surprendre. En effet, les réseaux neuronaux ont été utilisés depuis longtemps dans des problèmes difficiles de reconnaissance des formes (Mendel, 1970) et de classification. En fait, la conception de reconnaisseurs statistiques performants capables de minimiser la probabilité d'erreurs, et des méthodes d'apprentissage qui leur sont associées, est un problème récurrent en reconnaissance de formes. Le regain d'intérêt pour les réseaux neuronaux a replacé ce problème sur la sellette. Au-delà des différences apparentes entre les divers modèles proposés, deux aspects sont essentiels dans la conception d'un système de reconnaissance statistique : les fonctions utilisées pour décrire les frontières de classes et le critère d'apprentissage utilisé pour déterminer les paramètres de ces fonctions.

Dans un premier temps, nous analyserons les différents modèles neuromimétiques. Ensuite, nous détaillerons les étapes à suivre à utiliser pour distinguer des formes. Et enfin, nous développerons les différents types étudiés dans la reconnaissance de formes, en particulier les formes visuelles et auditives.

# I) Modèles neuromimétiques :

Dans cette partie, nous commencerons par expliquer le fonctionnement d'un neurone formel qui est un élément capital des quelques modèles que nous présenterons dans un second temps.

## 1) Le neurone formel

Un modèle neuromimétique ou réseau de neurones artificiels ou modèle connexionniste ou encore *Parallel Distributed Processing* (PDP) *model* (Rumelhart, 1986) est formé d'un grand nombre de cellules élémentaires simples, fortement interconnectées, le plus souvent du type illustré sur la figure 1.

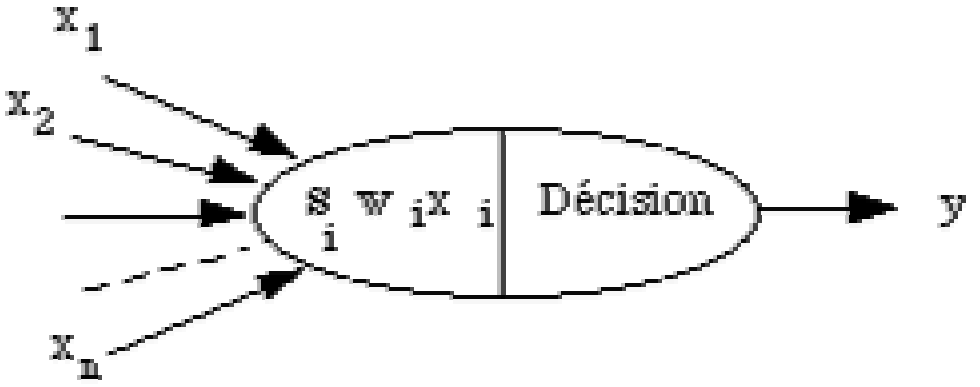



**Figure 1.** Principe d'un neurone artificiel

La cellule élémentaire est appelée "neurone" car son fonctionnement est fondé sur celui d'un automate proposé comme une approximation du fonctionnement du neurone biologique (McCullough, 1943). La sortie de cette cellule est une fonction non linéaire de la somme pondérée de ses entrées. Une forme analytique très courante pour la décision est la fonction sigmoïde, mais d'autres fonctions peuvent également être utilisées. La topologie du réseau, c'est-à-dire la façon dont les cellules sont interconnectées, est une caractéristique essentielle d'un tel réseau.

## 2) Modèles les plus courants

Parmi les modèles les plus utilisés, on peut citer les perceptrons, les réseaux récurrents et les cartes auto organisatrices.

Les perceptrons sont des réseaux sans contre-réaction : les sorties des neurones de l'ième couche forment les entrées de la (i+1)ème couche. La figure 2 montre la structure d'un perceptron à trois couches dont une couche cachée, appliqué à un problème de reconnaissance globale de mots. Ce modèle est issu des travaux de F. Rosenblatt sur le perceptron mono couche (Rosenblatt, 1962).

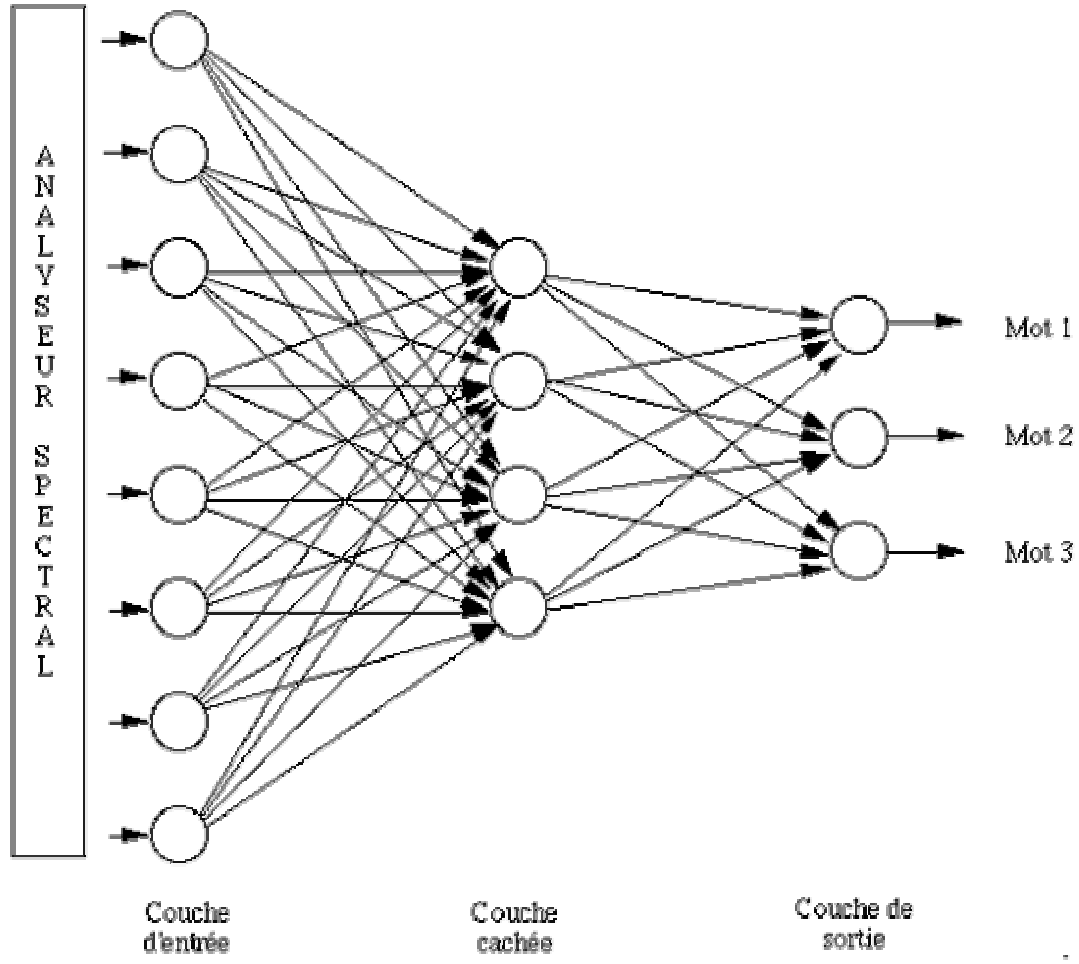



**Figure 2.** Structure d'un perceptron à trois couches

Les réseaux récurrents sont caractérisés par le fait que les entrées d'un neurone peuvent être aussi bien des entrées que des sorties d'autres neurones. Un exemple typique est donné par le modèle de Hopfield (Hopfield, 1982). Des réseaux récurrents simples, construits à partir de perceptrons, ont été largement étudiés en reconnaissance de la parole.

Le principe des cartes auto organisatrices, en particulier les cartes de Kohonen (Kohonen, 1982), est inspiré de l'organisation topologique du cortex des mammifères. Le modèle se compose de deux couches de neurones :

- une couche d'entrée,
- une couche représentant une carte de paramètres dont les cellules reçoivent des données à la fois de la couche d'entrée et de cellules voisines de cette deuxième couche. Cette connectivité latérale à l'intérieur de la couche est identique pour toutes les cellules et décroît avec la distance séparant deux cellules (interaction en forme de "chapeau mexicain"). Son effet est de concentrer l'activité du réseau en une petite région en réponse à la présentation d'un phonème ou d'une brève séquence de parole en entrée du réseau, et donc d'accentuer globalement le contraste dans le réseau.

Les modèles neuromimétiques possèdent des propriétés qui les rendent intéressants dans des domaines comme la reconnaissance de la parole, notamment:

- ils sont capables d'apprendre à partir d'exemples, dans un cadre d'apprentissage discriminant permettant d'améliorer la reconnaissance d'une classe de formes et, simultanément, le rejet des autres classes. De plus, les fonctions discriminantes apprises peuvent être fortement non linéaires. Ces modèles constituent ainsi une classe d'approximateurs très puissants de fonctions;
- ils ne nécessitent pas d'hypothèse forte sur les propriétés statistiques des données en entrée (contrairement aux modèles stochastiques de type modèles de Markov cachés MMC, en anglais HMM) ;
- ils présentent des structures parallèles régulières facilement implantables sur des matériels de grande performance (y compris sous forme de circuits VLSI spécifiques).

Il faut noter que l'exploitation des connaissances fournies par la psychologie et les neurosciences constitue une voie de recherche alternative prometteuse pour la conception de nouveaux modèles neuromimétiques. Le modèle de Kohonen, qui exploite les mécanismes d'inhibition latérale et les propriétés de phonotopie, en est un premier exemple. D'autres modèles ont également été proposés, en particulier permettant une gestion du temps interne au modèle.

## 3) Apprentissage dans les modèles neuromimétiques

L'apprentissage dans les modèles neuromimétiques se fait par présentation en entrée du réseau de formes représentatives de la population étudiée. Le résultat peut être la création de nouvelles connexions entre les cellules élémentaires, la modification des poids de ces connexions ou encore une topologie de réseau caractérisée par un arrangement particulier des cellules. Il existe deux grands types d'apprentissage :

- l'apprentissage non supervisé, dans lequel on se contente de présenter des formes sans indication sur leur identité. Une règle d'apprentissage est la règle de Hebb

(Hebb, 1949) qui revient à augmenter le poids de la connexion entre deux cellules si celles-ci sont simultanément actives et à le diminuer dans le cas contraire.

Les cartes de Kohonen utilisent un apprentissage non supervisé compétitif, inspiré de la loi de Hebb. Un autre exemple de modèle à apprentissage non supervisé est le modèle ART (Carpenter, 1988) ;

- l'apprentissage supervisé, dans lequel on impose une réponse en sortie du réseau pour une forme d'entrée donnée. Dans ce cas, il s'agit de calculer l'erreur commise par le réseau comme une fonction de la "distance" entre la sortie désirée et la sortie obtenue. Il existe deux grands types d'erreurs utilisés en pratique, correspondant à deux critères d'optimisation :

  - le critère des moindres carrés fondé sur l'utilisation d'une distance euclidienne entre les formes,
  - le critère d'entropie croisée dans lequel on considère les sorties du réseau comme des distributions de probabilités sur les variables d'entrée. Il s'agit alors de maximiser l'information mutuelle entre les sorties réelles et désirées.

Dans le cas du perceptron multicouches, l'algorithme d'apprentissage, dit de rétro propagation du gradient d'erreur, est itératif (Le Cun, 1985 ; Rumelhart, 1986). Le principe est d'adapter les différents poids des connexions du réseau à chaque présentation d'une forme en entrée, selon le gradient de l'erreur commise en sortie.

# II) Procédure à suivre pour la reconnaissance de forme :

On s'appuie sur le schéma classique d'un processus de reconnaissance de formes pour décrire les principaux traitements à effectuer et leurs objectifs.

Buts des étapes du schéma :

- Numérisation : obtenir une représentation des données à traiter qui soit manipulable en machine.
- Prétraitement : élimination des bruits, normalisation, re-échantillonnage, amélioration des contrastes, etc.
- Calcul des représentations : obtenir une représentation des données compatible avec les outils d'apprentissage et de décision utilisés.
- Apprentissage : à partir d'un ensemble d'exemplaires, construire une représentation des classes.
- Analyse : assigner une forme inconnue à une classe.
- Post traitement : valider les décisions de l'analyse sur la base de connaissances (du domaine).

Dans la pratique, un système de reconnaissance des formes s'éloigne souvent de ce schéma. Des traitements en amont sont souvent nécessaires pour isoler la forme à reconnaître de son contexte, ce qui est en soit est un problème de reconnaissance (segmentation forme/fond, délimitation d'une forme dans un ensemble). Des traitements ultérieurs sont aussi utiles pour valider les décisions et éventuellement les remettre en cause.

Le caractère séquentiel du processus de reconnaissance, tel qu'il est présenté dans ce schéma, n'est pas toujours la meilleure option. Ainsi, une erreur dans la segmentation de la forme à reconnaître augmente forcément le risque d'une mauvaise reconnaissance. Il est possible d'introduire une boucle dans le processus, remettant en cause la segmentation après analyse des résultats de la reconnaissance. Il est aussi possible de fusionner segmentation et reconnaissance. C'est le cas en écriture où la segmentation en lettres d'un mot cursif peut être couplée à la reconnaissance des suites de lettres formant un mot lexicalement valide (même principe pour la segmentation d'un signal de parole en phonèmes). Ce schéma illustre en fait l'immersion du noyau dur de la reconnaissance de formes (les méthodes d'apprentissage et d'assignation d'une donnée à une classe) dans un système qui, partant d'un signal brut, va permettre de résoudre un problème de reconnaissance.

# III) Les différentes formes de reconnaissance :

Comme on l'a vu précédemment, il y a 3 grands domaines pour « la forme » : la vision, la parole et le toucher. Nous allons surtout nous attarder sur les 2 premiers aspects.

## 1) Vision

Nous allons aborder trois activités : l'analyse d'objets (qui est de fait la méthodologie générale), la reconnaissance de caractères et enfin l'analyse de scènes en 3 dimensions (3D). Ces deux derniers domaines étant des applications directes de la première activité.

### a) Analyse d'objets

Détermination de l'objet, caractérisation, comparaison entre l'image connue et l'image vue au niveau du pixel. On peut considérer dans ce processus trois phases distinctes :

- prétraitement : simplification de l'image visualisée,
- extraction des primitives : caractéristiques principales de l'objet,
- classification.

L'opération de prétraitement englobe l'extraction des contours i.e. la détection de courbe continue, l'analyse de continuité, le seuillage qui correspond à choisir un niveau de gris, par exemple, tout pixel plus foncé sera codé par **1**, tout pixel plus clair étant codé par **0**.

L'extraction des primitives consiste à coder l'image au moyen de descripteurs pertinents; pertinence dépendant du type de traitement ultérieur. Par exemple on peut s'intéresser à la surface, au périmètre, au nombre de trous, à la dimension du rectangle circonscrit... On rajoute aussi des descripteurs de position et d'orientation comme le centre de gravitée, les angles...

La classification est en fait le processus de reconnaissance proprement dit, il s'agit de comparer l'image visualisée avec la connaissance du système. En général on utilise l'une des deux méthodes suivantes :

- méthode du plus proche voisin, c'est aussi la plus simple et la plus longue. Il faut comparer les objets un à un; ce qui est rendu d'autant plus difficile que les contraintes initiales soient lâches (objet arrivant dans n'importe quel sens, superposition possible...).
- méthode de décision par arbre binaire, cette technique demande de choisir un nombre optimal de critères discriminants (descripteurs pertinents), c'est la technique la plus couramment utilisée dans le cadre de la reconnaissance de caractères.

Les recherches menées en synthèse d'images sont faites en étroite collaboration avec l'analyse d'un objet digitalisé (problème de stockage, représentation de l'image...).

## b) Reconnaissance de caractères

Actuellement plusieurs types de lecteurs optiques sont disponibles sur le marché (lecteur code barre, OCR...) depuis ceux spécialisés dans la reconnaissance d'une ou plusieurs polices de caractères jusqu'aux lecteurs aptes à apprendre n'importe quoi.

Le premier lecteur optique avec reconnaissance de caractères date de 1955, les recherches sur le sujet furent abandonnées par les grosses sociétés (IBM, Control DATA) lorsqu'au début des années 60 parurent les premiers résultats sur la synthèse vocale (et le mythe de la ``paperless society''). Seuls quelques constructeurs spécialisés continuèrent à travailler sur le sujet (et heureusement vu l'état de l'art en synthèse vocale au seuil du troisième millénaire). En 1974, R. Kurzweil a l'idée de concevoir un lecteur optique permettant aux non-voyants de lire des textes classiques. La machine associe lecture optique et synthèse vocale pour restituer le texte original, en 1985 la KDEM 4000 est l'un des systèmes les plus performants.

La reconnaissance nécessite une squelettisation du caractère. Une fois simplifié, la représentation peut se faire en utilisant le code de Freeman basé sur le fait qu'un pixel possède au plus 8 voisins (numéroté de 0 à 7) permettant ainsi un codage sur un octet.

On peut aussi utiliser des descripteurs tels que verticale, horizontale, courbe... qui sont indépendants de la police de caractères utilisée, la reconnaissance pouvant alors s'effectuer à partir de masques partiels tels que ceux décrits dans la figure 3 ou par des heuristiques, comme la distribution aléatoire de droites dans le plan avec étude statistique du nombre d'intersection avec le caractère (cf. figure 3). Dans cette figure on remarque que la lettre « B » a plus d'intersection avec les droites que la lettre « A ».

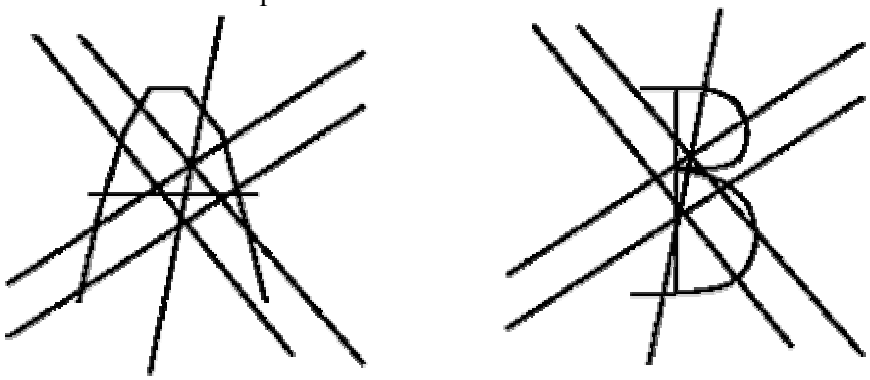


**Figure 3.** Droites aléatoires

### c) Analyse de scènes

La difficulté dans l'analyse de scènes 3D réside dans le fait que toute l'information n'est pas accessible directement problème de faces cachées, que l'on rencontre aussi en synthèse d'images.

On cherche dans ce cas à détecter les points caractéristiques, par exemple, dans le cas de volumes simples, les angles ou les jonctions de droites entre les différents contours. À partir de ces informations l'ordinateur (ou le robot) doit décomposer la scène en éléments simples.

## 2) Parole :

L'analyse de la reconnaissance vocale a donné lieu à de nombreux travaux depuis plus de 25 ans, les applications ne se restreignent pas uniquement aux ordinateurs. Déjà certains systèmes (simples voire simplistes) équipent des véhicules de haute gamme, on peut imaginer que d'ici peu les voitures réagiront à des ordres, que le téléphone pourrait composer automatiquement des numéros, dans ce cas on peut imaginer que la personne utilisant un composeur vocal associe un mot clef à un numéro donné, puis n'a plus qu'à citer ce mot, pour que le numéro soit appelé (la société Thomson CSF a d'ailleurs développé et fabriqué de tels composants). Le Katalavox de M. Kempf (1985) utilisé pour commander un appareil chirurgical à l'aide des quatre commandes directionnelles « up », « down », « left » et « right ».

Les modèles neuronaux ont été utilisés avec succès pour pré traiter le signal de parole et en extraire des paramètres caractéristiques.

La plupart des méthodes de traitement numérique de signal sont de type linéaire. Des méthodes non linéaires de filtrage adaptatif utilisant un perceptron multicouches ont été utilisées (Gibson, 1989 ; Knecht, 1995).

Les modèles neuronaux ont également été introduits pour réduire le niveau de bruit d'un signal vocal. L'idée est de considérer le processus d'amélioration d'un signal comme une transformation dans un espace paramétrique. Cette transformation peut être non linéaire, les modèles neuronaux étant capables d'apprendre à partir d'exemples des fonctions de grande complexité. Un perceptron multicouches a ainsi été utilisé pour effectuer une réduction de bruit dans le domaine temporel. Le système a été testé avec des auditeurs humains (Tamura, 1988 ; Tamura, 1990) et a fourni de bons résultats, mais au prix d'une complexité de calcul assez grande. Une autre méthode a été proposée dans (Sorensen, 1991). Elle consiste à utiliser un modèle d'oreille suivi par un réseau multicouche de réduction de bruit dans le domaine spectral. Des méthodes similaires à celles présentées ci-dessus peuvent être utilisées pour effectuer une adaptation acoustique à un nouveau locuteur.

Les modèles neuronaux peuvent également jouer un rôle dans l'extraction de paramètres pour la reconnaissance, principalement selon deux axes :

- l'extraction au niveau des couches cachées ou de la sortie de perceptrons (Elman, 1988) (Bourlard, 1988). Les paramètres ainsi obtenus sont des combinaisons des

données fournies en entrée du perceptron, à l'image des méthodes classiques de l'analyse de données,

- l'utilisation de modèles neuronaux à auto-organisation, capables d'extraire, par apprentissage non supervisé compétitif, des paramètres pertinents à partir de corpus de données brutes. Un exemple est celui des cartes de Kohonen (Kohonen, 1984). Ces cartes peuvent également servir à effectuer une quantification vectorielle du signal acoustique. L'extraction d'indices à la manière d'une analyse en composantes principales (Oja, 1982) a été effectuée avec un réseau muni de liens inhibiteurs en cascade entre les neurones de sortie (Rubner, 1990). La méthode a été appliquée à la parole par Beaugé, en 1993.

Une autre application intéressante est la séparation de sources acoustiques, mais aussi radar ou sonar. La solution présentée par Jutten, en 1991, est une solution adaptative qui s'inspire de la réalité du contrôle neurobiologique.

# IV) En résumé :

On retrouve ainsi les principales reconnaissances des formes telle que l'analyse d'images vue comme un ensemble de données afin d'y retrouver des objets ou des configurations prédéterminées, la reconnaissance de caractère à partir d'une image afin d'y récupérer des caractères en particulier des lettres, un texte typographié et numérisé, ou un manuscrit, et la reconnaissance de la parole consistant généralement en l'analyse du son pour en séparer les différentes syllabes prononcées et ainsi en ressortir la diction.

Les principales étapes d'une application de reconnaissance de forme sont le recueil des données brutes, la génération de caractéristiques, la sélection des caractéristiques pertinentes et l'évaluation du système. On peut donc rencontrer les problèmes suivants : la décision, l'apprentissage, l'évaluation des performances, la prise en compte du temps et la fusion des données hétérogènes.

Par conséquent, afin de mettre en œuvre ces reconnaissances de formes les modèles utilisés sont les modèle neuromimétiques. L'objectif visé est la mise en exergue d'une organisation structurelle des neurones. Chaque structure est dotée d'une fonction particulière et ces structures adaptent leur comportement par des mécanismes d'apprentissage. L'apprentissage implique des modifications physiques des connexions entre neurones. L'association entre plusieurs structures neuronales, dotées chacune d'une fonction précise, permet l'émergence d'une fonction d'ordre supérieure pour l'ensemble.

Dans leur grande majorité, les applications développées pour la reconnaissance de formes telle qu'elle soit sont complètement neuronales. A cela plusieurs raisons, la plus mauvaise est que le développeur ne dispose pas de compétences autres que connexionnistes pour envisager un couplage avec des techniques plus classiques. Une seconde raison est qu'une application purement connexionniste peut être interprétée plus avantageusement en terme de possibilités neuronales. Enfin, et c'est la principale, coupler les techniques classiques avec les techniques connexionnistes est un difficile problème.

Bien que les recherches en reconnaissance de formes aient beaucoup évolué ces dernières années, la complexité de ce domaine de recherche fait qu'il reste encore à améliorer et découvrir des techniques de reconnaissance.

# Conclusion :

L'intelligence artificielle sert-elle à quelque chose ? Oui, sans aucun doute, dans le cadre de la validation, scientifique ou appliquée, de conclusions. Quant aux statistiques qui en sont issues, sont incontestablement les seules à être non asymptotiques, donc véritablement valables. L'intelligence artificielle nous amènera-t-elle des super-intelligences comme dans les films… pourquoi pas, mais à ce niveau-là, sans doute les biotechnologies nous surprendront-elles en premier.

La plus grande partie des recherches en intelligence vise d'abord à comprendre un certain nombre de phénomènes, à tester des hypothèses et à faire des expériences par le biais de simulations informatiques. Or l'appréciation de la justesse d'une simulation est très subjective et c'est pourquoi l'intelligence artificielle est loin de prétendre à la même rigueur que les mathématiques ou la physique. L'intelligence artificielle, comme son histoire l'a amplement montré depuis sa naissance, fait plutôt partie de ce que l'on a coutume d'appeler la "techno-science", c'est-à-dire ce passage obligé où la science mesure ces impacts technologiques à l'aune des impératifs sociaux.

Nous constatons que les méthodes expliquées dans la première partie (la reconnaissance des problèmes) sont importantes car on les retrouve dans tous les domaines que touche l'intelligence artificielle et notamment elles sont appliquées dans les parties 2 et 3 (stratégie de jeux et reconnaissance de forme).

Au jour d'aujourd'hui, l'intelligence artificielle est très employée dans les jeux vidéos, ces jeux prennent une grande place sur le marché des ventes et de la production, les consommateurs ont des goûts variés mais on retrouve une intelligence artificielle dans tous les jeux qu'ils achètent. Les jeux touchent un large public d'âge divers et varié.

L'importance d'une telle réussite peut venir du fait que les jeux vidéos et l'intelligence artificielle développée dans ceux-ci ont subi une évolution importante. Les jeux accrochent les joueurs car on est plongé dedans et il devient de plus en plus intéressant de se confronter à un adversaire de plus en plus intelligent.