

# Module de Doctorat 3<sup>ème</sup> Cycle LMD

## Les systèmes complexes avancés

Présenté par : Prof. Krolladi Mohamed-Khireddine  
Département d'Informatique  
Facultés des Sciences Exactes  
Université Echahid Hamma Lakhdar d'El Oued  
Tél. 0770314924  
Email. krolladi@univ-eloued.dz et krolladi@yahoo.fr  
Site Web. www.univ-eloued.dz  
<http://krolladi.doomby.com/> et <http://krolladi.e-monsite.com/>



## Les systèmes complexes avancés

### Syllabus

#### a - Les objectifs du cours

Ce cours est un état de l'art sur les systèmes complexes. On va voir les bases des systèmes complexes naturels et artificiels, et leurs éventuelles applications. On verra que les systèmes complexes possèdent leurs propres démarches du point de vue résolution des problèmes. On notera que l'objectif est de fournir à l'étudiant les approches évoluées pour la modélisation des systèmes complexes.

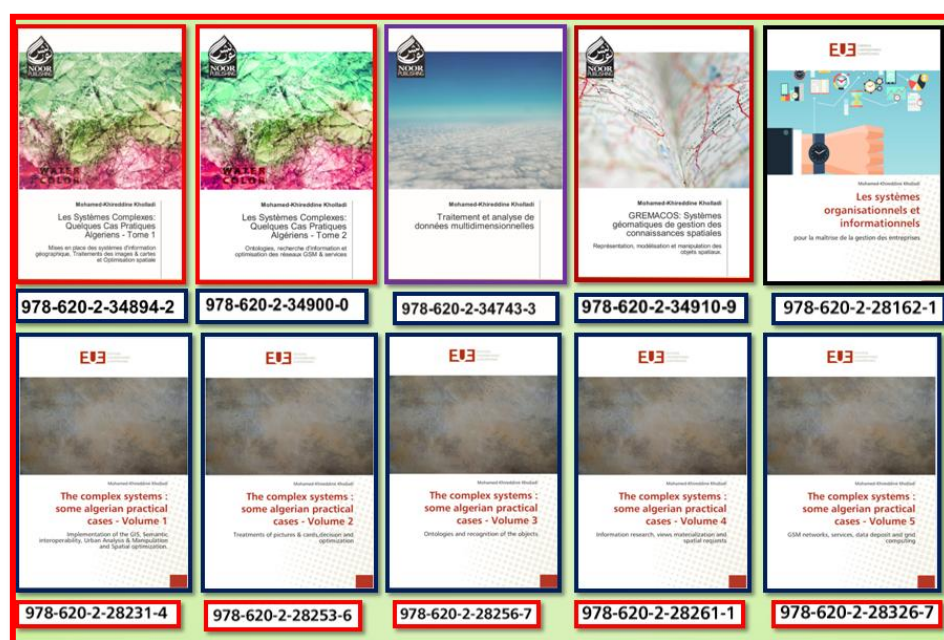
#### b - Les ouvrages recommandés

1. Z. Michalewics, "Genetic Algorithms + Data Structures = Evolution Programs", Springer, 1996.
2. A.E Eiben, J.E Smith: "Introduction to Evolutionary Computing", Springer, 2003.
3. W.M Spears: "Evolutionary Algorithms: The Role of Mutation and Recombination", Springer, 2004.
4. Melanie Mitchell (2009): Complexity: A guided Tour. Oxford University Press.
5. Claudios Gros (2010): Complex and Adaptive Dynamical Systems. Second Edition, Springer.
6. Dan Braha, Ali Minai, Yaneer Bar-Yam (2010): Complex Engineered Systems: Science Meets Technology. Springer.
7. Jay Xiong (2011), New Software Engineering Paradigm Based on Complexity Science: an introduction to NSE, Springer.

- A. Bonnet, "l'intelligence artificielle: promesses et réalités", InterEditions, Paris, 1984, 271 pages.
8. H. Farreny, "Les Systèmes experts: principes et exemples", Technique avancée de l'informatique, CEPADUEES-EDITIONS, Toulouse, 1985, 256 pages.
- A. Faure, "Perception et reconnaissance des formes", éditestes, Paris, 1985, 286 pages.
9. M-K. Kholadi, "Traitement et analyse de données multidimensionnelles", 220 pages, Noor Publishing, Schaltungsdienst Lange o.H.G, Berlin, 20 Février 2018, ISBN-13 : 978-620-2-34743-3, ISBN-10 : 6202347430, [www.morebooks.fr](http://www.morebooks.fr), [www.get-morebooks.com](http://www.get-morebooks.com) ou sur / in Amazon.
10. M-K. Kholadi, "Les systèmes complexes : quelques cas pratiques algériens – Tome 1 : mises en place des systèmes d'information géographique, traitements des images & cartes et optimisation spatiale", 600 pages, Noor Publishing, Schaltungsdienst Lange o.H.G, Berlin, 20 Février 2018, ISBN-13 : 978-620-2-34894-2, ISBN-10 : 6202348941, [www.morebooks.fr](http://www.morebooks.fr), [www.get-morebooks.com](http://www.get-morebooks.com) ou sur / in Amazon.
11. M-K. Kholadi, "Les systèmes complexes : quelques cas pratiques algériens – Tome 2 : ontologies, recherche d'information et optimisation des réseaux GSM & services", 532 pages, Noor Publishing, Schaltungsdienst Lange o.H.G, Berlin, 22 Février 2018, ISBN-13 : 978-620-2-34900-0, ISBN-10 : 620234900X, [www.morebooks.fr](http://www.morebooks.fr), [www.get-morebooks.com](http://www.get-morebooks.com) ou sur / in Amazon.
12. M-K. Kholadi, "GREMACOS : systèmes géomatiques de gestion des connaissances spatiales – représentation, modélisation et manipulation des objets spatiaux", 296 pages, Noor Publishing, Schaltungsdienst Lange o.H.G, Berlin, 26 Février 2018, ISBN-13 : 978-620-2-34910-9, ISBN-10 : 6202349107, [www.morebooks.fr](http://www.morebooks.fr), [www.get-morebooks.com](http://www.get-morebooks.com) ou sur / in Amazon.
13. M-K. Kholadi, "Les systèmes organisationnels et informationnels pour la maîtrise de la gestion des entreprises", 241 pages, EUE : Editions universitaires européennes, Schaltungsdienst Lange o.H.G, Berlin, 15 Juin 2018, ISBN-13 : 978-620-2-28162-1, ISBN-10 : 6202281626.
14. M-K. Kholadi, "The complex systems : some algerian practical cases (Volume I) – implementation of the geographical information systems, semantic interoperability, urban analysis & manipulation and spatail optimization", 552 pages, EUE : Editions universitaires européennes, Schaltungsdienst Lange o.H.G, Berlin, 22 Février 2018, ISBN-13 : 978-620-2-28231-4, ISBN-10 : 6202282312, [www.morebooks.fr](http://www.morebooks.fr), [www.get-morebooks.com](http://www.get-morebooks.com) ou sur / in Amazon.
-

15. M-K. Kholadi, “The complex systems : some algerian practical cases (Volume II) – treatment of pictures & cards, decision and optimization”, 412 pages, EUE : Editions universitaires européennes, Schaltungsdienst Lange o.H.G, Berlin, 22 Février 2018, ISBN-13 : 978-620-2-28253-6 ISBN-10 : 6202282533, [www.morebooks.fr](http://www.morebooks.fr), [www.get-morebooks.com](http://www.get-morebooks.com) ou sur / in Amazon.
16. M-K. Kholadi, “The complex systems : some algerian practical cases (Volume III) – Ontologies and recognition of the objects”, 416 pages, EUE : Editions universitaires européennes, Schaltungsdienst Lange o.H.G, Berlin, 26 Février 2018, ISBN-13 : 978-620-2-28256-7 ISBN-10 : 6202282568, [www.morebooks.fr](http://www.morebooks.fr), [www.get-morebooks.com](http://www.get-morebooks.com) ou sur / in Amazon.
17. M-K. Kholadi, “The complex systems : some algerian practical cases (Volume IV) – information research, views materialization and spatial requests”, 440 pages, EUE : Editions universitaires européennes, Schaltungsdienst Lange o.H.G, Berlin, 26 Février 2018, ISBN-13 : 978-620-2-28261-1, ISBN-10 : 6202282614, [www.morebooks.fr](http://www.morebooks.fr), [www.get-morebooks.com](http://www.get-morebooks.com) ou sur / in Amazon.
18. M-K. Kholadi, “The complex systems : some algerian practical cases (Volume V) – GSM networks, services, data deposit and grid computing”, 320 pages, EUE : Editions universitaires européennes, Schaltungsdienst Lange o.H.G, Berlin, 26 Février 2018, ISBN-13 : 978-620-2-28326-7 ISBN-10 : 6202283262, [www.morebooks.fr](http://www.morebooks.fr), [www.get-morebooks.com](http://www.get-morebooks.com) ou sur / in Amazon.

### c -Les livres du Prof. Kholadi Mohamed-Khireddine



## **d -Contenu du module**

- Etat de l'art sur les systèmes complexes
- Les systèmes complexes naturels
- Les systèmes complexes artificiels
- Les applications
- Les approches classiques
- Les algorithmes génétiques
- Les automates cellulaires et quantiques

## **Sommaire**

### I - Qu'est-ce que l'intelligence artificielle ?

#### I.1 - Définitions

##### I.1.1 - Intelligence

##### b - Connaissance

##### c - Raisonnement

##### d – Formalisme de représentation de connaissances

##### e – Typologie des problèmes

##### f – Résolution de problème

##### g – Définition opérationnelle d'intelligence artificielle (essai de Turing)

#### I.2 - Objectifs

### II - Histoire de l'intelligence artificielle

#### II.1 – La gestation de l'intelligence artificielle

#### II.2 – La naissance de l'intelligence artificielle (en 1956)

#### II.3 – Bien avant enthousiasme, bonnes attentes (de 1952 à 1969)

#### II.4 – Une dose de la réalité (de 1956 à 1969)

#### II.5 – Les systèmes à bases de connaissances (de 1969 à 1979)

#### II.6 – A partir de 1980, l'intelligence artificielle devient une industrie

#### II.7 – A partir de 1986, le retour des réseaux de neurones

#### II.8 – A partir de 1987, l'intelligence artificielle devient une science

### III - Les systèmes complexes

#### III.1 - Introduction rapide de la vue d'ensemble des principaux travaux

##### III.1.1 – Définition formelle d'un problème

##### III.1.2 – Réseaux de neurones artificiels

##### III.1.3 – Programme de développement des systèmes intelligents

III.1.4 – Automates cellulaires (CA)

III.2 – Pourquoi parler de systèmes complexes ?

III.3 – Modélisation des systèmes complexes

III.4 – Outils SysML pour la modélisation des systèmes complexes

IV - Résolution des problèmes par l'exploration

IV.1 - Introduction rapide de la vue d'ensemble

IV.2 -Exploration de cartes

IV.3 - Cube Rubik en 3\*3\*3

IV.4 - Huit-puzzle ou taquin à huit pièces

IV.5 - N-Reines

IV.6 - Missionnaires et cannibales

IV.7 - Problème de la rivière

V - Algorithmes de base d'exploration

V.1 - Quels sont les critères utilisés pour comparer les différentes stratégies d'exploration?

V.2 - Les complexités du temps et de l'espace?

V.3 -Facteurs de branchement pour certains problèmes

V.4 - Un exemple de jeu : les vacances roumaines

V.5 - La mise en œuvre du générique de l'algorithme d'exploration

V.6 - Principaux aspects de l'algorithme d'exploration sur l'état de l'espace

VI - Stratégies d'exploration non informées (aveugles)

VI.1 - Algorithmes d'exploration de base

VI.2 - Comparaison des algorithmes d'exploration

VII – Stratégies d'exploration informées (heuristiques)

VII.1 - Utilisation des connaissances spécifiques du problème pour aider à l'exploration

VII.2 - Plus formellement, lesquelles fonctions heuristiques travaillent?

VII.3 - Algorithmes d'exploration de base

VII.4 - Fonctions heuristiques

VIII – Les conclusions

IX - Émergence

IX.1 - La résolution des problèmes par émergence : éco-résolution

IX.2 - Huit-puzzle – Taquin : le principe

IX.3 - Taquin & algorithmique

IX.4 - Éco-résolution : les principes

IX.5 - Éco-résolution : l'algorithme général

IX.6 - Taquin & éco-résolution

X - Développement des systèmes de programmes intelligents

X.1 - Approches symboliques - les systèmes à base de connaissances (SBC)

- a - Les systèmes experts
- b - Présentation d'une rapide introduction
- c - Les premiers systèmes experts

X.2 - CLIPS (C Système de production intégré de langue)

X.3 - Les langages de programmation de l'intelligence artificielle

- a - LISP (Traitement de listes)
- b - PROLOG (programmation logique)
- c - Comparaison des manipulations symboliques et les techniques traditionnelles

XI – Approche connexionniste

XI.1 - Apprentissage machine : réseaux de neurones

XII - Nouvelle Intelligence Artificielle - Les systèmes complexes

- XII.1 - Vie artificielle et systèmes complexes
- XII.2 - Vie artificielle et automates cellulaires
- XII.3 - Automates cellulaires - le jeu de la vie

XIII - Nouveaux algorithmes de tri basé sur le calcul naturel

- XIII.1 - Algorithme traditionnel de tri
- XIII.2 - Tri des boules: Une nouvelle façon pour le tri sur la base de l'informatique naturelle
- XIII.3 - Boules-Tri étendu
- XIII.4 - Boules-tri-AC mise en œuvre
- XIII.5 - Boules-tri-AC étendu - mise en œuvre

XIV - L'intelligence collective

- XIV.1 - Comment placer n agents en cercle de manière équi-répartie?
- XIV.2 - Implémentation de NetLogo

XV – Les conclusions

XVI – Les références

XVII - Cas n°1: Système complexe SIG pour le suivi de la remontée des eaux de la wilaya d'El-Oued Souf

- XVII.1 – La présentation de l'interface utilisateur
- XVII.2 - La couche "Population"
- XVII.3 - La couche "Puits"
- XVII.4 - La couche "Salinité"

- XVII.5 – La couche "Nitrates dissous"
- XVII.6 – La couche "Nappe phréatique"
- XVII.7 – Les Requêtes
  - XVII.7.1 – La requête "Couche de population"
  - XVII.7.2 – La requête "Couche de puits"
  - XVII.7.3 – Les représentations graphiques
- XVIII – Le cas n°2 : Système complexe SIG pour l'étude de l'évolution de la répartition de la population de la wilaya d'Adrar
  - XVIII.1 – Le modèle conceptuel
  - XVIII.2 – La présentation de l'environnement de développement
  - XVIII.3 – La présentation de l'interface utilisateur
  - XVIII.4 – les manipulations des données géographiques de population et réalisation
    - XVIII.4.1 – Les requêtes descriptives d'identification et de caractérisation
    - XVIII.4.2 – Les requêtes descriptives d'identification et de caractérisation
    - XVIII.4.3 – Les requêtes spatiales
    - XVIII.4.4 – La conclusion et les perspectives
- XIX – Le cas n°3: Système complexe SIG pour le suivi et la gestion des données de la géologie minière de la région de Constantine
  - XIX.1 – Le découpage en processus de l'application suivi de la géologie minière
  - XIX.2 - La modélisation du système informatique
  - XIX.3 – Le diagramme de contexte
  - XIX.4 – La liste préliminaire des cas d'utilisation
  - XIX.5 - Le diagramme des classes
  - XIX.6 – La réalisation
    - XIX.6.1 – La fenêtre "Information"
    - XIX.6.2 – La fenêtre "Mot de passe"
    - XIX.6.3 – La fenêtre de travail (principale)
    - XIX.6.4 – La fenêtre "Boîte de message d'erreur"
    - XIX.6.5 – Les requêtes
  - XIX.7 – La conclusion et les perspectives
- XX – La conclusion générale de la présentation des applications en Algérie

## **I - Qu'est-ce-que l'intelligence artificielle?**

### **I.1 – Définitions**

#### **I.1.1 - Intelligence**

L'intelligence est définie comme la capacité d'acquérir et de mettre en œuvre des connaissances et la faculté de la pensée et du raisonnement. L'intelligence artificielle est l'étude des systèmes complexes qui agissent d'une manière qui semble être intelligente pour n'importe quel observateur humain. L'intelligence artificielle implique le suivi des méthodes basées sur le comportement intelligent des humains et d'autres animaux pour résoudre des problèmes très complexes. L'intelligence artificielle est concernée par les problèmes réels, difficiles et, qui exigent des processus compliqués et la connaissance complexes et sophistiqués du raisonnement. "L'intelligence artificielle est l'étude des idées qui permettent à des ordinateurs d'être intelligents" selon P. Winston. "C'est la science et la technologie de faire des machines intelligentes, particulièrement des programmes informatiques intelligents. On le lie aux tâches semblables utilisant des ordinateurs pour comprendre l'intelligence humaine, mais l'intelligence artificielle ne doit pas se confiner aux méthodes qui sont biologiquement observables" selon John McCarthy, On distingue deux familles d'intelligence artificielle, les unes sont dites faibles et les autres sont dites fortes. Pour la première, il s'agit de l'intelligence artificielle, qui développe des applications utiles et puissantes. Pour la seconde, il s'agit les machines à raisonnement, qui ont des esprits cognitifs comparables aux humains. Dans ce cours, on va s'occuper pour l'essentiel de l'intelligence artificielle faible.

#### **I.1.2 - La connaissance**

Quelle est la différence entre les trois termes : donnée, information et connaissance?

1. La donnée transporte l'information : ce sont des signaux non interprétés ;
2. L'information est une interprétation de la donnée ;
3. La connaissance utilise l'information dans le cadre d'actions, dans un but précis. Les actions peuvent être la prise de décisions, la création de nouvelles informations, etc.

Questions fondamentales

1. Acquisition de la connaissance du domaine
2. Représentation de la connaissance (un formalisme syntaxique + sémantique)
3. Contrôle du raisonnement (complétude et validité)



La connaissance, c'est la capacité à mobiliser des informations pour agir. Le passage de l'information à la connaissance est lié à l'expérience de l'action, cela veut dire qu'il n'a pas de frontière parfaitement définie.

## **a - Définition**

La connaissance, c'est l'information (donnée) qui influence un processus. La connaissance, c'est l'ensemble des notions et des principes qu'une personne acquiert par l'étude, l'observation ou l'expérience et qu'elle peut intégrer à des habiletés. Le synonyme de génie cognitif : discipline étudiant l'extraction et la formalisation de connaissances provenant d'un expert humain en vue d'automatiser le raisonnement. En informatique, l'ingénierie des connaissances évoquerait les techniques pour manipuler des connaissances sur ordinateur.

### **I.1.3 – Raisonnement**

Le raisonnement a pour préoccupation la modélisation et l'automatisation de processus de raisonnement et de prise de décision, dans une perspective d'assister l'utilisateur ou le décideur. Ils concernent les problématiques suivantes :

- La modélisation des croyances et leur dynamique, et le raisonnement automatisé ;
- Modèles logico-mathématiques pour la modélisation des croyances incomplètes, incertaines et/ou partiellement incohérentes ;
- Révision des croyances ;
- Raisonnement sur l'action et la causalité ;
- Fusion de croyances ;
- Apprentissage automatisé.

Le raisonnement est un "processus cognitif" qui permet d'obtenir de nouveaux résultats ou bien de vérifier la réalité d'un fait en faisant appel soit à différentes lois, soit à des expériences, quelque soit leur domaine d'application : mathématiques, système judiciaire, physique, santé, pédagogie, etc. Les mathématiques sont un langage, pour pouvoir s'exprimer rigoureusement, adapté aux phénomènes complexes, qui rend les calculs exacts et vérifiables. Le raisonnement est le moyen de valider ou d'infirmer une hypothèse et de l'expliquer aux autres. On dit que l'individu effectue des inférences et que le mécanisme d'élaboration de ces inférences s'appelle le raisonnement.

La compétence en capacité de raisonnement est la capacité à trouver et évaluer des renseignements ainsi que la capacité à identifier et analyser un problème, choisir une solution et évaluer l'impact de celle-ci. Elle consiste à résoudre des problèmes, de prendre des décisions éclairées, d'user (ou de faire preuve ?) de pensée critique, d'organiser et planifier son travail, d'utiliser sa mémoire dans un but spécifique et enfin de rechercher des renseignements.

### I.1.4- Formalismes de représentation de connaissances

Il existe plusieurs formalismes. Le choix d'un formalisme à utiliser dépend à la fois du domaine d'application, des opérations à mettre en œuvre sur ces connaissances (type de problème à résoudre) et surtout de la culture du concepteur du modèle de représentation comme sur la figure I.1.

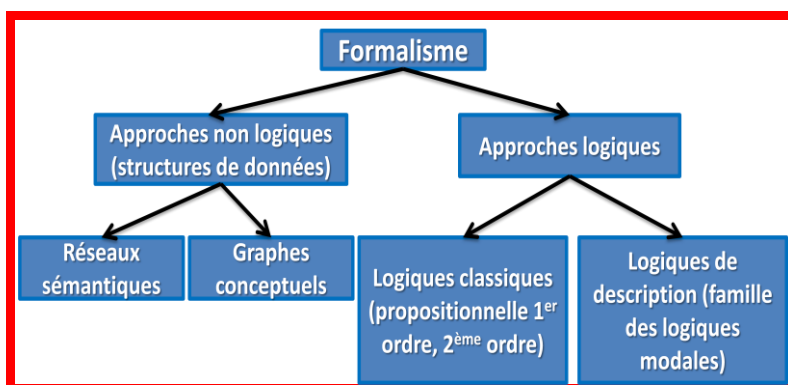


Figure I.1 - Formalismes de représentation de connaissances

### I.1.5 - Typologie des problèmes

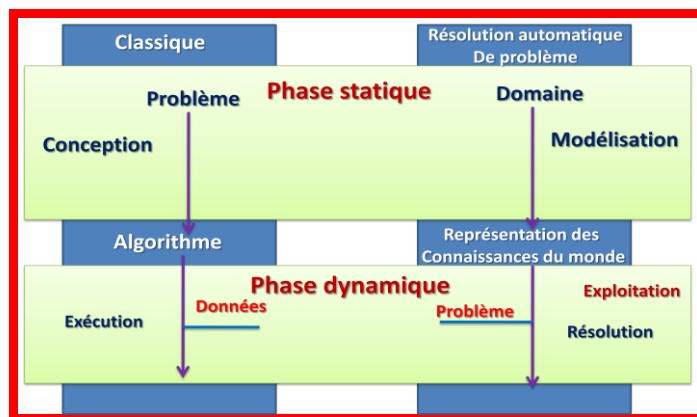
On distingue quatre types de typologie des problèmes :

1. Les problèmes de tournées étroitement apparentés aux problèmes de détermination de trajets. Ils présentent une différence notable. Cela consiste à visiter au moins toutes les villes en commençant et en terminant par la ville de départ.
2. Le problème du voyageur de commerce ou TSP (Traveling Salesperson Problem) est une variante du problème précédent dans lequel chaque ville doit être visitée exactement une fois. Le but de départ est de chercher l'itinéraire le plus court et au moindre coût.
3. Le problème d'agencement des circuits VLSI ou celui de placement-routage nécessite de positionner des millions de composants et de connexions sur une puce de façon à minimiser l'encombrement, les temps morts et les capacités parasites.

4. Le problème de déplacement d'un robot est une généralisation du problème de recherche d'itinéraire. Au lieu de suivre des chemins discrets, un robot peut se mouvoir dans un espace continu, caractérisé par un ensemble infini d'actions et d'états possibles.

### I.1.6 - Résolution de problème

La figure I.2 illustre les deux démarches de résolution d'un problème. La démarche traite le problème pour le résoudre et la démarche de résolution traite le domaine pour le modéliser à des de représentation des connaissances.



*Figure I.2 – Résolution de problème*

### I.1.7 - Définition opérationnelle d'intelligence artificielle (Essai de Turing)

En 1950, Alan Turing a proposé une définition opérationnelle d'intelligence en employant un essai composé :

1. D'un interrogateur (une personne qui posera des questions) ;
2. D'un ordinateur (une machine intelligente) ;
3. D'une personne (un expert humain) qui répondra aux questions ;
4. Et d'un rideau (un séparateur).

L'ordinateur passe le test d'intelligence, si un humain, après avoir posé quelques questions écrites, ne peut pas dire si les réponses étaient d'une personne ou pas comme sur la figure I.3.



*Figure I.3 – Le test de Turing*

Pour pouvoir répondre automatiquement, l'ordinateur devrait posséder les outils suivants :

- Traitement de langage naturel : pour communiquer avec succès.
- Représentation des connaissances : pour stocker ce qu'il sait ou entend.
- Raisonnement automatique : pour répondre à des questions et tirer des conclusions en utilisant les informations stockées.
- Étude de machine : pour s'adapter à de nouvelles circonstances et détecter et extrapoler des modèles.
- Vision d'ordinateur : pour percevoir des objets.
- Robotique : pour manœuvrer des objets et à les déplacer.

## I.2 - Objectifs

Les objectifs sont de créer des systèmes (logiciels ou machines) intelligents, qui pensent, réfléchissent et raisonnent comme des humains, et/ou pensent, réfléchissent, raisonnent rationnellement, et/ou se comportent, agissent et réagissent comme les humains, et/ou se comportent, agissent et réagissent rationnellement.

Le domaine de l'intelligence artificielle est influencé par plusieurs disciplines :

1. Informatique, génie (comment programmer et implanter l'IA?);
2. Mathématiques, statistiques (limites théoriques de l'IA?);
3. Neurosciences (comment le cerveau fonctionne?);
4. Psychologie cognitive (comment l'humain réfléchit?);
5. Économie, théorie de la décision (comment prendre une décision rationnelle?);
6. Linguistique (quelle est la relation entre le langage et la pensée?);
7. Philosophie (quel est le lien entre le cerveau et l'esprit?);
8. Etc.

L'intelligence artificielle a commencé comme tentative de comprendre la nature de l'intelligence, mais elle s'est développée dans des champs scientifiques et technologiques affectant beaucoup d'aspects du commerce et des sociétés. Les objectifs principaux de l'intelligence artificielle sont :

### **a - Dans le domaine de l'ingénierie :**

- Il s'agit de résoudre les réels problèmes en utilisant les connaissances et le raisonnement.

- L'intelligence artificielle peut nous aider à résoudre des problèmes difficiles et réels, en créant de nouvelles occasions dans les affaires, l'ingénierie et dans beaucoup d'autres champs d'application.

### **b - Dans le domaine scientifique :**

- Utiliser les ordinateurs comme plate-forme pour étudier l'intelligence elle-même.
- Des théories de conception scientifique présumant des aspects d'intelligence alors ils peuvent mettre en application ces théories sur un ordinateur.

Même pendant que la technologie d'intelligence artificielle devient intégrée dans le tissu de la vie quotidienne, les chercheurs d'intelligence artificielle restent concentrés sur les grands défis d'automatiser l'intelligence humaine.

### **I.3 – Problèmes avec des solutions algorithmiques**

Les programmes informatiques conventionnels résolvent généralement des problèmes ayant des solutions algorithmiques. Les langues algorithmiques incluent C, Java, C#, etc. L'intelligence artificielle traite des problèmes NP-complets (les problèmes complexes) avec des langues classiques d'intelligence artificielle incluent le LISP et le PROLOG.

### **I.4 – Les domaines de l'intelligence artificielle**

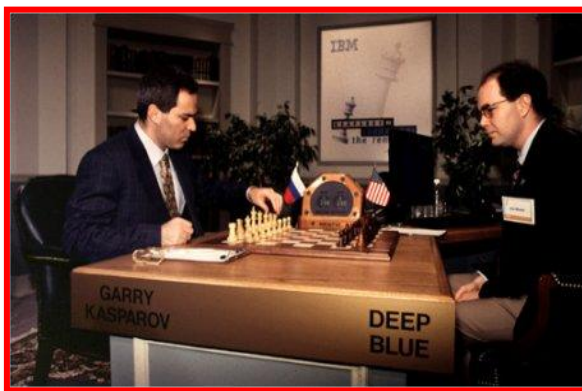
Ils sont divers et variés comme suit :

1. Compréhension de la parole
2. Reconnaissance des formes
3. Sens tactile en Robotique
4. Compréhension des langues naturelles
5. Mécanismes de raisonnement
6. Réalité augmentée et virtuelle
7. Systèmes experts
8. Réseaux de neurones
9. Vie artificielle et biotechnologie
10. Télédétection et séismologie
11. Réseaux de transports
12. Réseaux sémantiques
13. Enseignement assisté par ordinateur
14. Système multi agents

15. Jeux
16. Informatique et imagerie médicale
17. Traitement et Synthèse d'image
18. Etc.

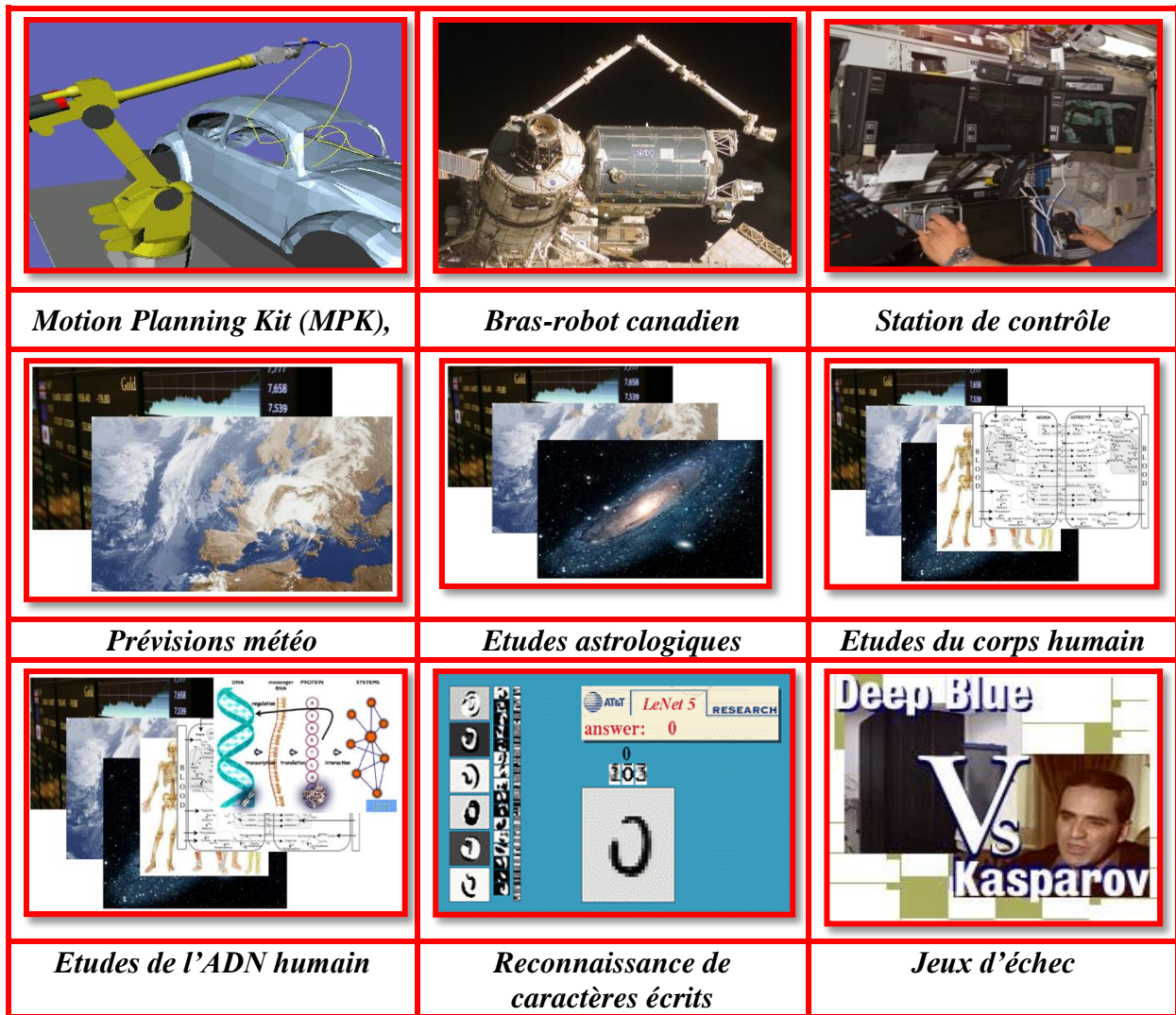
## **I.5 – Les exemples d'applications**

L'intelligence artificielle dans les jeux comme Deep Blue d'IBM qui est le premier programme à battre un champion mondial d'échec Mr Garry Kasparov et gagne une partie en 1996, gagne une série de 6 parties en 1997. Il pouvait prévoir entre six et vingt mouvements de suite comme sur la figure I.4. Mr Garry Kasparov a dit qu'il avait senti une nouvelle sorte d'intelligence chez son adversaire. On peut considérer que l'ordinateur est maintenant le champion du monde aux échecs.



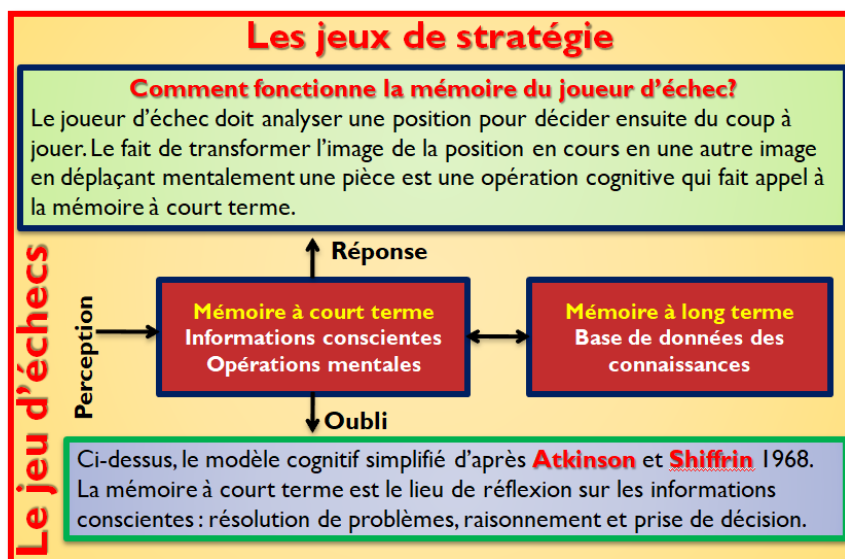
*Figure I.4 – La machine Deep Blue d'IBM contre le champion du monde d'échec*

L'intelligence artificielle dans la planification de trajectoires pour un corps articulé avec évitement d'obstacles comme sur la figure I.5. On a par exemple, la machine robot Motion Planning Kit (MPK) Jean-Claude Latombe et Mitul Saha à Stanford University, le bras-robot canadien et la station de contrôle ou dans la gestion des systèmes dynamiques complexes, où on a des interactions locales simples et un comportement global complexe comme les prévisions météo, les études de l'espace sidérale en astrologie, les études du corps humain et de l'ADN et même dans le domaine des jeux tels que Deep Blue d'IBM. Elle a aussi investi des jeux de stratégie TDGammon, le joueur de jacquet champion du monde, construit par Gerry Tesauro de recherche d'IBM, le championnat d'échecs entre la machine Deep Blue et le champion du monde des échecs Mr Gary Kasparov et le programme chinois des contrôleurs intelligents.



*Figure I.5 – Machines robots et systèmes dynamiques*

La figure I.6 illustre le jeu de stratégie d'échecs.



*Figure I.6 – Stratégie de jeu d'échecs*

Ce jeu de stratégie nécessite une mémoire à court terme et une mémoire à long terme. Et surtout de savoir comment fonctionne la mémoire d'un joueur d'échecs. Dans la même idée, la figure I.7 traite un exemple de problème à résoudre en trois coups.

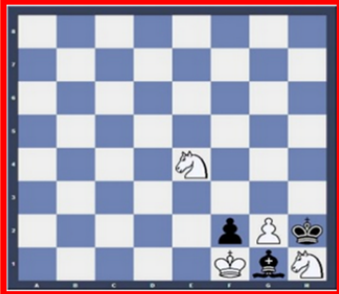
**Les jeux de stratégie**

Un exemple de problème à résoudre : mat en 3 coups

**Le jeu d'échecs**

C'est à partir de la représentation de la position dans la mémoire à court terme que le joueur d'échecs établit son jugement, définit le plan stratégique et procède aux calculs tactiques.

Les Blancs jouent et font mat en trois (03) coups.



La mémoire peut fusionner des éléments en les associant les uns aux autres de façon à ne former qu'un seul élément. Le nouvel item formé renferme autant d'informations que tous les autres réunis. Ce procédé est appelé emboitage (chunking en anglais).

*Figure I.7 – Exemple de problème à résoudre – mat en trois coups*

La figure I.8 illustre l'importance de la reconnaissance de patterns pour améliorer les techniques de simulation et l'expertise des joueurs dans le jeu, où les nombreuses connaissances et leurs interconnexions sont mises à rude contribution.

La reconnaissance de patterns est déterminante

**Le jeu d'échecs**

Les simulations de la mémoire experte, conduites pour reproduire les performances de professionnels, ont montré que ces derniers disposent en mémoire d'environ 300.000 chunks. Cela laisse méditatif sur la qualité de connaissances qui sont stockées en mémoire procédurale (coups tactiques, manœuvres typiques) et verbale (stratégies typiques, plans « maison ») des champions d'échecs.



La force du maître vient dans un premier temps de ces nombreuses connaissances et de leurs interconnexions. Ainsi, dans certaines positions complexes pour l'amateur, le maître trouve son chemin rapidement, simplement parce que la reconnaissance de patterns (configuration de pièces) active en lui des procédures et des solutions déjà étudiées et stockées dans la mémoire à long terme (les chunks).

*Figure I.8 – La reconnaissance de patterns*

La figure I.9 illustre les entraînements efficaces d'un joueur d'échecs.



**Les jeux de stratégie**

**Les entraînements efficaces pour un joueur d'échecs**

**Le jeu d'échecs**

Dans le cadre d'un programme d'entraînement, il faut nécessairement limiter le nombre de concepts à apprendre en fonction de la durée et du nombre de séances. La situation idéale est le rattachement d'une information nouvelle à une connaissance déjà intégrée.



L'empan mnésique est la limite du nombre d'items que l'on peut conserver simultanément en mémoire est une autre caractéristique de la mémoire qui a un fort impact sur l'apprentissage.

*Figure I.9 – Les entraînements efficaces pour un joueur d'échecs*

La figure I.10 illustre la démarche de la méthode de progression aux échecs.

**Les jeux de stratégie**

**Méthode pour progresser aux échecs**

**Le jeu d'échecs**

La première étape consiste à établir un diagnostic de son propre jeu ce qui n'est pas toujours chose facile. À la difficulté d'être objectif quant à ses propres décisions, vient parfois s'ajouter la difficulté de se replonger dans ses défaites. Cependant, il ne faut limiter l'analyse de ses parties aux défaites. L'impasse n'a pas lieu d'être, et les victoires devraient aussi être passées au crible. L'assistance d'un logiciel est appréciable pour détecter ce qui échappe à l'œil humain.

Phase	Type de faute	Compteur
Ouverture	Méconnaissance du plan	
	Motifs tactiques typiques	
Milieu de jeu	Planification	
	Motifs tactiques	
Finale	Technique	
	Exploitation de l'avantage	
	Stratégie	

Tableau pour compter les types de fautes et savoir ce qu'il faut travailler en priorité.

*Figure I.10 – La méthode pour progresser aux échecs*

La figure I.11 illustre la méthode pour réaliser un diagnostic exhaustif. Ici, on a une liste exhaustive de toutes les erreurs possibles dans le jeu d'échecs.



Figure I.11 – Méthode pour réaliser un diagnostic exhaustif

La figure I.12 illustre le choix du bon coup aux échecs.

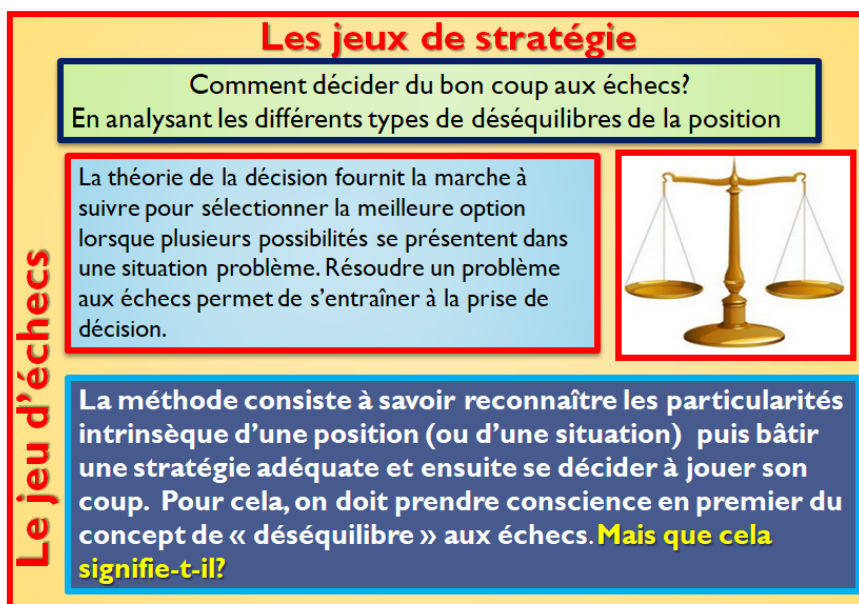


Figure I.12 – Choix du bon coup aux échecs

La figure I.13 illustre les différents types de déséquilibres dans une position aux échecs. Les grands maîtres des échecs les listent au nombre de huit.

**Les jeux de stratégie**

Quels sont les différents types de déséquilibres dans une position aux échecs?

**Le jeu d'échecs**

Voici un catalogue détaillé des divers déséquilibres pouvant survenir dans une position aux échecs. Cette liste a été établie par le maître international et grand pédagogue des échecs **Jeremy Silman** :

1. Roi adverse exposé
2. Avantage matériel
3. Meilleure structure de pion
4. Contrôle d'une ligne ou d'une case clé
5. Pièces mieux placées
6. Avantage d'espace
7. Avance de développement
8. Initiative



*Figure I.13 – Les différents types de déséquilibres dans une position aux échecs*


La figure I.14 illustre la technique de réflexion structurée en cinq points.

**Les jeux de stratégie**

La technique de réflexion structurée

**Le jeu d'échecs**

1. Identifier les déséquilibres positifs et négatifs des deux (02) camps;
2. Décider du secteur de l'échiquier vers lequel se porteront les efforts. On ne peut intervenir qu'à l'endroit où existe un déséquilibre favorable ou la possibilité d'en provoquer un;
3. Ne calculez aucune variante! Imaginez plutôt diverses positions de rêve auxquelles on aimerait pouvoir arriver.
4. Après avoir trouvé une position de rêve qui vous convient, voit s'il existe une façon pour la matérialiser. Si on se rend compte que le rêve est impossible à réaliser, imagine-en un autre plus accessible,
5. On peut enfin faire la liste des coups qu'on désire calculer - on les appelle les coups candidats. Ces coups sont tous ceux susceptibles de mener à la position de rêve, etc.



*Figure I.14 – La technique de réflexion structurée en cinq points*

Dans le domaine de la compréhension du langage naturel, les traducteurs d'intelligence artificielle parlés impriment, ce qu'on veut dans des langues étrangère et de la compréhension du langage naturel, les contrôleurs de vocabulaire et les contrôleurs de grammaire.

Dans les systèmes experts, le premier système expert, DENDRAL (Dendritic Algorithm), contient les connaissances des experts chimistes et les utilise pour répondre aux problèmes d'identification de composants chimiques à partir de mesures physiques (spectroscopie de masse, etc.). Dans la géologie, le système expert de prospection minière

PROSPECTOR porte l'évaluation du potentiel minéral de l'emplacement géologique ou de la région.

Dans la prise de décision financière, les fournisseurs des cartes de crédit, banques, systèmes de l'utilisation de l'intelligence artificielle des sociétés de prêt hypothécaire pour détecter la fraude et pour accélérer des transactions financières. Dans le matériel et logiciel de configuration, les systèmes d'intelligence artificielle configurent l'ordinateur fait sur commande, les communications, et les systèmes de fabrication, garantissant le temps d'installation maximum d'efficacité et de minimum d'acheteur.

Dans les systèmes diagnostiques, le diagnostiqueur, un système de diagnostic médical (suggère des essais et fait le diagnostic) s'est développé par Heckerman et toutes autres recherches de Microsoft. Le système de MYCIN pour diagnostiquer des infections bactériennes du sang et suggérer des traitements.

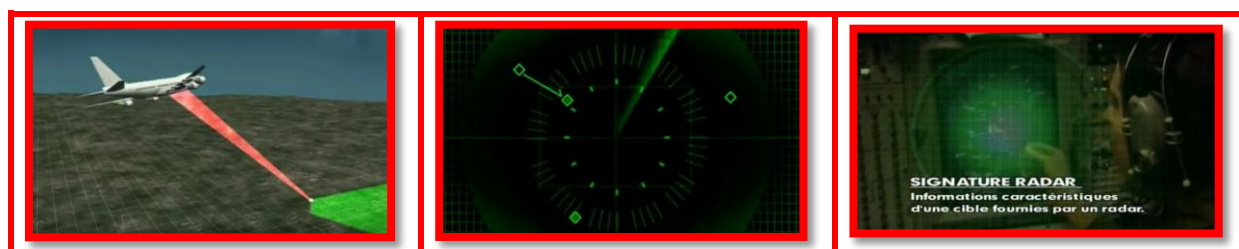
La figure I.15 illustre les progrès dans le domaine de la robotique, où le robot effectue des mouvements d'agilité et de déplacement. La vidéo nous renseigne sur ces mouvements. L'augmentation importante de la robotique dans divers secteurs tels que : les jeux, pour manipuler des conditions dangereuses et pour réaliser les travaux pénibles entre autres, par des exemples de voitures automatisées, de joueur de ping-pong, d'exploitation, de construction, d'agriculture et de collectes d'ordures.



*Figure I.15 - Robotique*

Dans la technologie de la furtivité, les radars sont devenus de véritables outils de détection des intrus dans un espace aérien d'un pays. Il en est de même pour les sonars pour la détection des attaques sous marines. Un radar émet des signaux et reçoit des échos des appareils survolant une région donnée. Depuis le radar, on peut déterminer les caractéristiques

de taille, de forme, de direction et de vitesse de la cible aérienne que l'on appelle la signature radar. Tout le défi est comment concevoir des avions capables de diminuer sa perception par les radars ou les sonars, c'est-à-dire augmenter sa furtivité comme sur la figure I.16.



*Figure I.16 – La signature radar*

La forme de l'avion est dictée par les caractéristiques de furtivité comme le F117 américains, où l'aspect aérodynamique a été négligé, c'est-à-dire :

1. Réduction de la signature radar;
2. Peinture noir spécifique diminuant sa perception visuelle dans le ciel par l'absorption des ondes radars;
3. Réduction de tous les facteurs de détection de l'avion : infrarouge, visibilité, système de détection thermique de l'échappement, etc.

Aujourd'hui, des avions furtives avec une aérodynamique spécifique peut être des armes stratégiques comme les nouveaux bombardiers B20 et les chasseurs F22 Raptor USA (ou Sukhoï 37 russe) qui volent hors portée des radars jusqu'à 85000 pieds avec des missiles autonomes, intelligents et puissants (ARM) comme sur la figure I.17.



*Figure I.17 – Les avions furtives*

Le développement des radars doppler et la détection par interférence des perturbations du maillage des ondes des réseaux mobiles de téléphonie peuvent rendre visible ce qui est invisible comme sur l'image la vue d'un F117 américain. La stratégie militaire de placer plusieurs stations radars mobiles permettent par triangulation de détecter les avions invisibles. La furtivité n'est plus un élément de surprise dans un conflit. La recherche maintenant se déplace sur la conception des avions capables de voler hors portée des radars à plus de 85000

pieds et de tirer des missiles intelligents à partir de cette distance guidés par des satellites comme sur la figure I.18.



*Figure I.18 – Le développement des radars doppler*

On peut d'autres exemples, tels que l'identification d'écriture pour les lecteurs de code postal de service postal, la preuve automatisée de théorème pour l'utiliser les méthodes d'inférences pour prouver de nouveaux théorèmes et les moteurs de recherche de Web.

## **II - Histoire de l'intelligence artificielle**

### **II.1 - La gestation de l'intelligence artificielle**

- 1943            Mc Culloch et Pitts : Modèle de circuit booléen du cerveau
- 1950            Alan Turing : Turing's "la machine de calcul et d'intelligence"
- 1950s            Early AI programs, including Samuel's checkers program, Newell & Simon's Logic Theorist, Gelernter's Geometry Engine

### **II.2 - La naissance de l'intelligence artificielle (en 1956)**

- 1956            McCarthy organise la réunion de Dartmouth et inclut Minsky, Shannon, Newell, Samuel, Simon

Le nom "intelligence artificielle" fût adopté.

### **II.3 - Bien avant enthousiasme, bonnes attentes (de 1952 à 1969) :**

- 1957            Le solveur général de problèmes [Newell, Simon, Shaw @ CMU]
- 1958            Création du laboratoire d'Intelligence artificielle du MIT par Minsky et McCarthy
- 1958            LISP, [McCarthy], deuxième langage de haut niveau (MIT AI Memo 1)

- 1963           Création du laboratoire d'intelligence artificielle de Stanford par McCarthy  
1965           L'algorithme complet de raisonnement logique de Robinson

#### **II.4 - Une dose de la réalité (de 1966 à 1973) :**

- 1966-74       Intelligence artificielle à la découverte de la complexité informatique  
1966-72       Shakey, Le robot mobile de SRI [Fikes, Nilson]

#### **II.5 - Les systèmes à bases de connaissances (de 1969 à 1979)**

- 1969           Publication de "Perceptrons" [Minsky & Papert], La recherche de réseau de neurones disparaît presque  
1969-79       Développement précoce des systèmes à bases de connaissances  
1970           SHRDLU, Système de langage naturel de Winograd  
1971           MACSYMA, un système algébrique de manipulation symbolique

#### **II.6 - À partir de 1980, l'intelligence artificielle devient une industrie**

- 1980-88       Booms de l'industrie des systèmes experts  
1981           Japon: projet de cinquième génération, US: Microelectronics and Computer Technology Corp. Et UK: Alvey

#### **II.7 - À partir de 1986, le retour des réseaux de neurones**

- 1988-93       Poussé des systèmes experts : "hiver de l'intelligence artificielle"  
1985-95       Retour à la popularité des réseaux de neurones  
1986-91       Système GREMACOS du Pr Kholadi Mohamed-Khireddine au laboratoire LISI de l'INSA de Lyon sous la direction du Pr Robert Laurini.

#### **II.8 - À partir de 1987, l'intelligence artificielle devient une science**

- 1988           Réapparition des méthodes probabilistes et de théories de la décision, Théorie de l'apprentissage informatique, "Nouvelle intelligence artificielle " : ALife, gaz, calcul doux, calcul émergent, etc.  
1991-2018     Les systèmes complexes : quelques cas pratiques algériens du Pr Kholadi Mohamed-Khireddine, Laboratoire MISC

### III – Les systèmes complexes

#### III.1 - Introduction rapide de la vue d'ensemble des principaux travaux

##### III.1.1 - Définition formelle d'un problème

Dans le contexte de notre cours, un problème sera défini comme suit :

1. Un état initial;
2. Un ensemble d'actions;
3. Une fonction de successeur, qui définit l'état résultant de l'exécution d'une action dans un état;
4. Un ensemble d'états buts;
5. Et une fonction de coût, associant à chaque action un nombre non-négative (le coût de l'action).

On peut voir un problème comme un graphe orienté où les nœuds sont des états accessibles depuis l'état initial et où les arcs sont des actions. On appellera ce graphe l'espace des états. Une solution sera un chemin de l'état initial à un état but. On dit qu'une solution est optimale si la somme des coûts des actions du chemin est minimale parmi toutes les solutions du problème. La résolution des problèmes par l'exploration (explorations non guidées ou informées, explorations heuristiques, etc.). Les systèmes à bases de connaissances (systèmes experts, CLIPS, Prolog, LISP, etc.). Les machines à apprentissage (réseaux de neurones, RL (ou apprentissage par renforcement), etc.). Les systèmes complexes "intelligence artificielle moderne" (automates cellulaires, Gas, NNs, PSO, etc.).

##### III.1.2 - Réseaux de neurones artificiels

La figure III.1 symbolise la pensée voir les réseaux de neurones avec le site d'animation.



*Figure III.1 – Symbole de réseaux de neurones avec le site d'animation*



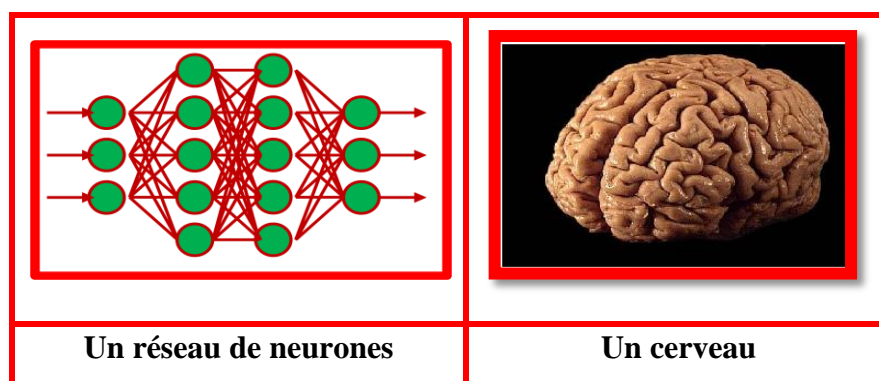
En 1982, les travaux du physicien John Hopfield relancent les recherches sur les réseaux de neurones. Hopfield propose un modèle de réseau particulier (totalement connecté) et un nouveau principe d'apprentissage, la rétro propagation du gradient, qui permet de résoudre le problème soulevé par Minsky et Papert. Mais d'autres problèmes encore plus ardues demeurent. De nombreuses applications industrielles voient le jour : diagnostic (voiture Renault, photocopieurs Canon, etc.), compression de données (JVC), prévision de consommation d'eau (Algérienne des Eaux "SEACO", entreprise nationale des barrages ENB, etc.), d'électricité (Sonelgaz), de trafic routier (Inforoute) ou de cours boursiers (Bourse d'Alger), reconnaissances des formes, etc. Par exemple, les machines de tri du courrier utilisent des réseaux de neurones formels pour reconnaître les chiffres des codes postaux sur les enveloppes comme sur la figure III.2.



*Figure III.2 – Machine de tri postal*

### III.1.3 - Programme de développement des systèmes intelligents

Les réseaux de neurones artificiels sont des tentatives brutes de modéliser fortement d'une manière massive et parallèle des traitements distribués que l'on suppose fait par le cerveau humain comme sur la figure III.3.



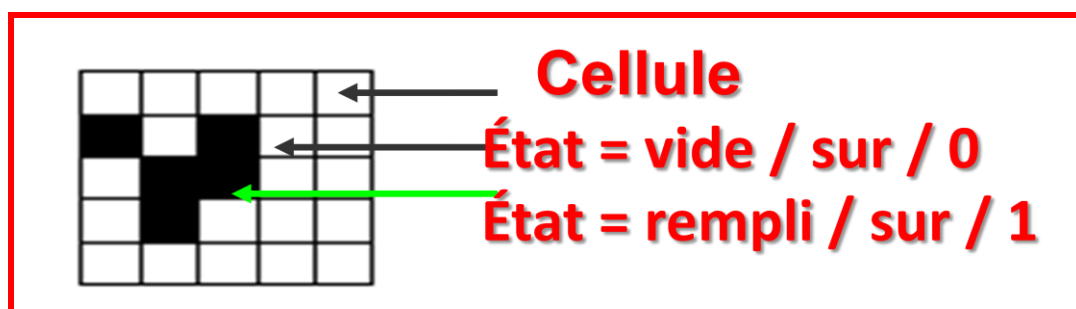
*Figure III.3 – Un réseau de neurone et un cerveau*

Au départ, Les réseaux de neurones ont intéressé deux secteurs principaux d'activités : la biologique avec des tentatives de modéliser des systèmes neuraux biologiques et l'informatique avec le développement des applications puissantes. Ils peuvent être employés pour répondre aux préoccupations suivantes de :

- Reconnaissance des structures : Cette image contient-elle un visage ?
- Problèmes de classification : Cette cellule est-elle défectueuse ?
- Prévisions : Donné ces symptômes, le patient a la maladie X.
- Prévisions ; Comportement de prévisions des marchés boursiers.
- Écriture : Le caractère est-il identifié ?
- Optimisation : Trouver le chemin le plus court pour le TSP.

### III.1.4 - Automates cellulaires (AC)

L'espace des automates cellulaires (AC) est un treillis des cellules (habituellement 1D, 2D, 3D) avec une géométrie particulière comme sur la figure III.4. Chaque cellule contient une variable d'une gamme limitée des valeurs (par exemple, 0 et 1). Toutes les cellules se mettent à jour de manière synchrone. Toutes les cellules emploient la même règle de mise à jour, dépendant seulement des relations locales et avancées de temps dans des étapes discrètes.



*Figure III.4 – L'espace des automates cellulaires*

#### a - Jeu de la vie : Automates cellulaires 2D utilisant des règles simples

La figure III.5 illustre le fonctionnement des automates cellulaires 2D, qui utilisent des règles simples expliquées dans le tableau associé à la figure. Le tableau donne les règles de fonctionnement ou de transition des cellules. C'est ce qu'on la vie de Conway.

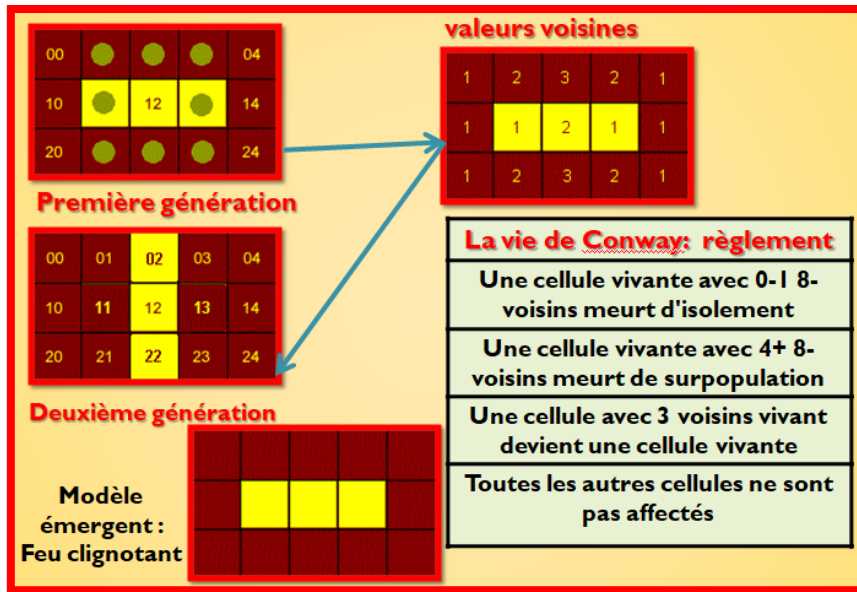


Figure III.5 – Jeu de la vie avec automates cellulaires 2D

## b - Jeu de la vie : les modèles émergents

La figure III.6 illustre le fonctionnement des automates cellulaires 2D, qui utilisent des modèles émergents qui expliquent le mode de transition des cellules dans le tableau associé à la figure. Le tableau donne les règles de fonctionnement ou de transition des cellules. C'est ce qu'on appelle aussi la vie de Conway.

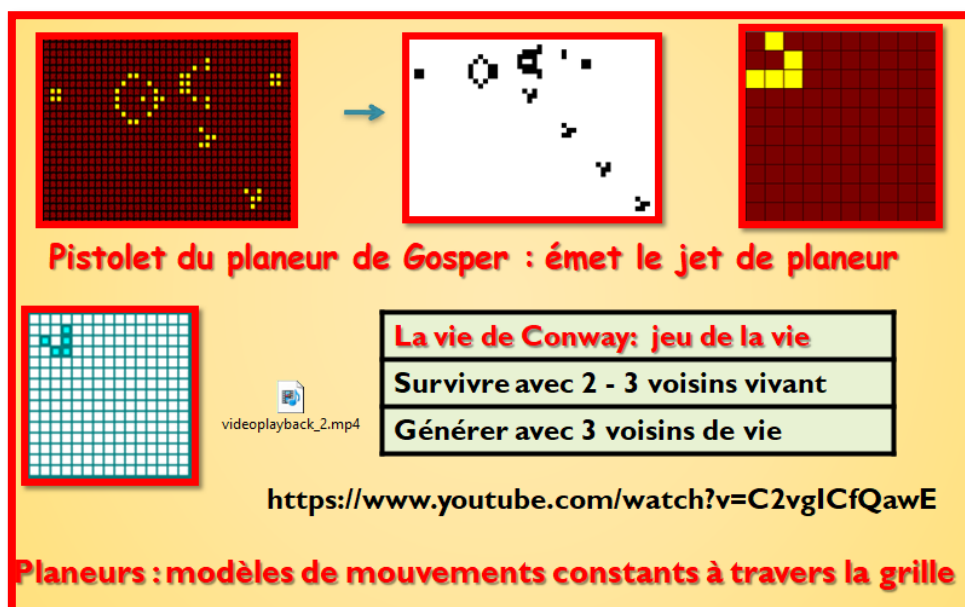
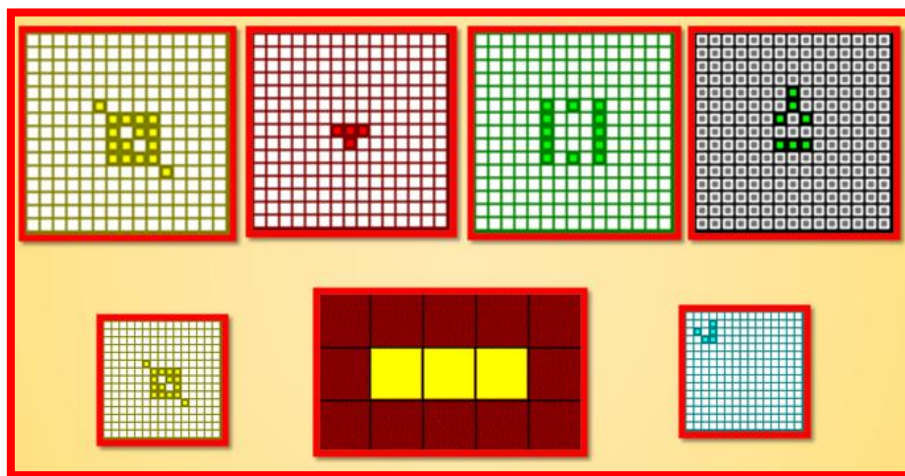


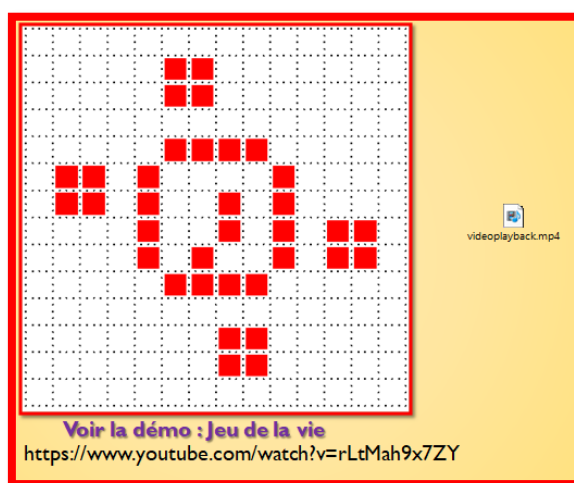
Figure III.6 – Jeu de la vie avec des modèles émergents

La figure III.7 illustre d'autres modèles émergents pour simuler le fonctionnement ou la transition des cellules.



*Figure III.7. D'autres modèles émergents*

La figure III.8 illustre le modèle émergent d'une montre avec le site de la démonstration de la vidéo.



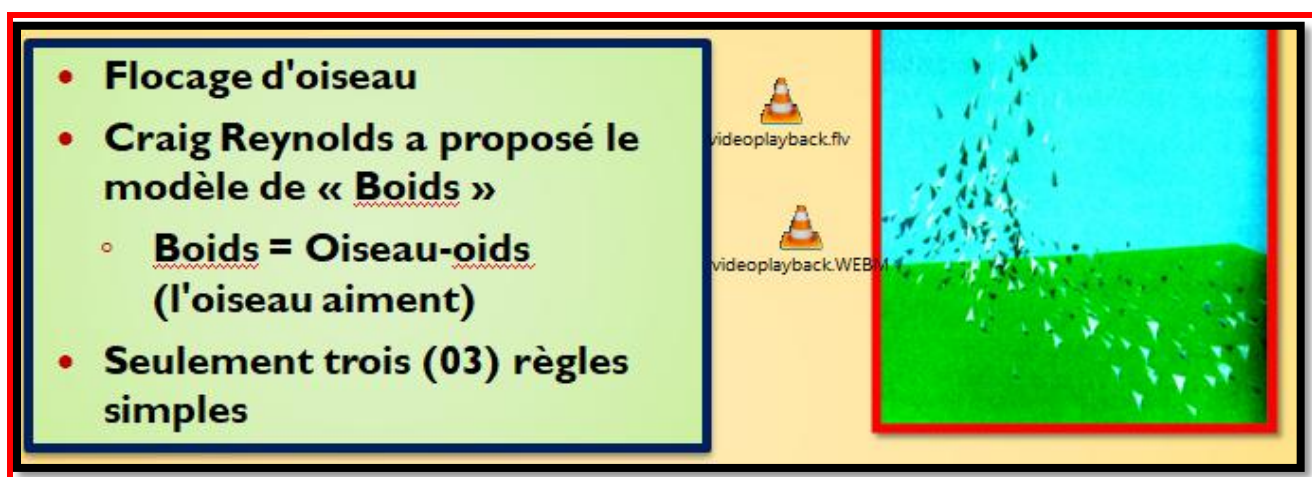
*Figure III.8 – Modèle émergent d'une montre*

### **c - La vie artificielle**

Une tentative de mieux comprendre la vraie modélisation des systèmes biologiques de la vie des entités dont on se rend compte. Les propriétés dans la vie de l'émergent motivent des scientifiques et divers explorateurs de la possibilité de créer artificiellement la vie et d'attendre l'inattendu. Quand quelques propriétés d'un émergent choisis deviennent plus que la somme de ses parties.

## c.1 - Boids de Craig Reynolds

Boids est un simulateur de robots virtuels comme un oiseau, appelés Boids, ainsi que la création d'un comportement émergent de flockage. Inspiré par le papier "cheptels, troupeaux, et les écoles : un modèle comportemental Distribué" par Craig Reynolds, il a mis en place un simulateur de Boids rapide, capable de simuler et de rendre plus de mille Boids au temps réel. La figure III.9 illustre le comportement d'oiseau par le modèle des Boids avec deux démonstrations en vidéo.



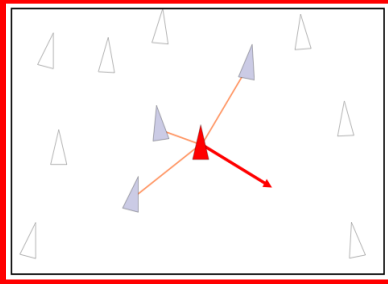
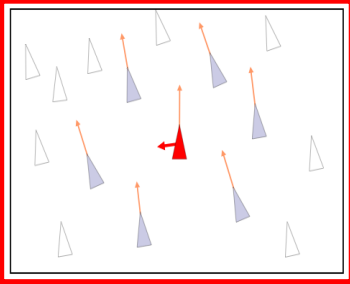
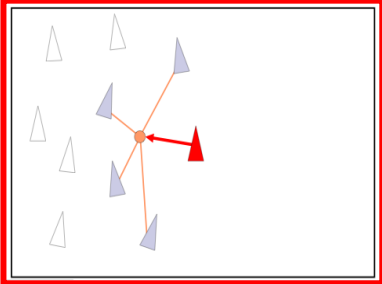
*Figure III.9 – Boids de Craig Reynolds*

Les règles simples provoquent le comportement complexe comme sur la figure III.10.



*Figure III.10 – Règles simples de comportement*

Figure III.11 illustre les règles d'évitement des obstacles pour l'anticollision entre les membres d'oiseaux, d'assortiment de vitesse par rapport aux autres membres et enfin de centrage de bande pour la cohésion du groupe d'oiseaux.

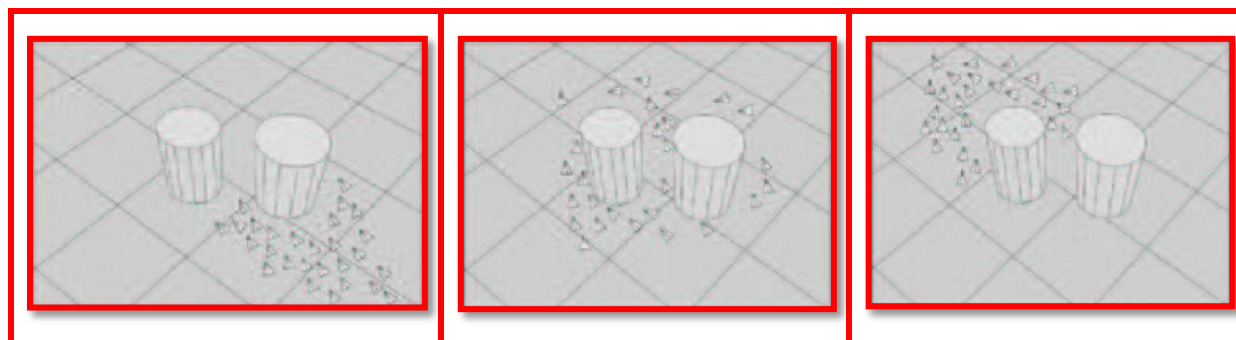
Évitement anticollision	Assortiment de vitesse : Alignement	Centrage de bande : Cohésion
		
<p align="center"><b>Règle 1</b></p> <p>Éviter la collision avec des oiseaux voisins (la séparation)</p>	<p align="center"><b>Règle 2</b></p> <p>Assortir la vitesse des oiseaux voisins (l'alignement)</p>	<p align="center"><b>Règle 3</b></p> <p>Séjour près des oiseaux voisins (cohésion)</p>

*Figure III.11 – Règles d'évitement, d'assortiment de vitesse et de centrage de la bande*

Les règles simples pour chacun des individus. Il n'y a aucune commande centrale, c'est-à-dire décentralisé et par conséquent robuste. Il y a apparition d'un système composé des oiseaux discrets effectuant des mouvements globaux coordonnés.

### c.2 - Apparition dans les systèmes complexes

La figure III.12 illustre le comportement des Boids de Craig Reynolds pour éviter les obstacles, dans notre cas les deux colonnes.



*Figure III.12 – Boids de Craig Reynolds avec évitement d'obstacles*

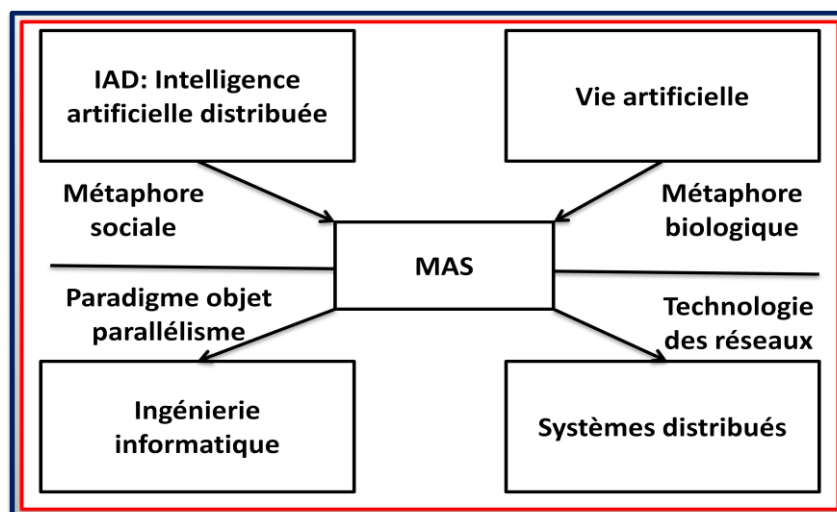
## III.2 – Pourquoi parler de systèmes complexes

On parle de systèmes complexes, quand on a un grand nombre de composantes, des relations complexes entre ces composantes, un haut degré de dynamique et des comportements

internes difficiles à analyser. Ce qui implique un fort débat entre le réductionnisme et les systèmes complexes.

### III.3 – Modélisation des systèmes complexes

C'est surtout le monde de la modélisation informatique comme sur la figure III.13.



*Figure III.13 – Le monde de la modélisation informatique*

### III.4 - Outils SysML pour la modélisation des systèmes complexes

SysML est un langage très adapté à la description des systèmes complexes techniques. Les anciens outils (FAST, SADT, diagramme des prestations (bête à corne), diagramme des inter-acteurs (pieuvre) et Grafcet) ont vécu. Leurs origines diverses sont très limitées (structure ou fonction / mécanique ou électronique ou ...) ne permet plus de bien modéliser un système actuel d'où le besoin d'un outil plus moderne (SysML basé sur UML). Les systèmes actuels sont de plus en plus intégrés et communicants. Ils offrent de plus en plus de fonctionnalités et la structuration classique est de moins en moins adaptée.

## IV- Résolution des problèmes par l'exploration

### IV.1 - Introduction rapide de la vue d'ensemble

#### Pourquoi explorer ?

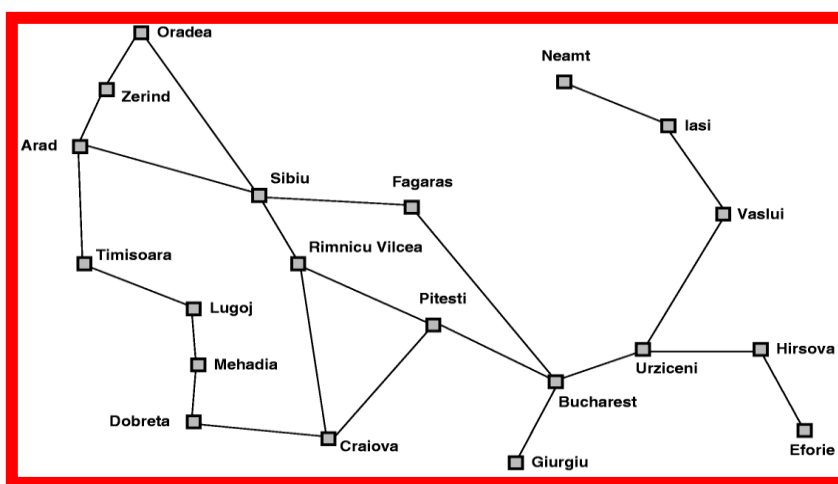
Les premiers travaux de l'intelligence artificielle étaient tournés principalement vers la preuve des théorèmes, la solution des puzzles et les jeux. Toute l'intelligence artificielle était basée sur l'exploration ou la recherche : pas totalement vrai (évidemment) mais plus vrai que L'on ne pourrait le penser et surtout trouver une bonne ou une meilleure solution à un problème parmi beaucoup de solutions possibles (voir la figure IV.1).



*Figure IV.1 – Les jeux d'échecs et de cube rubik en 3\*3\*3*

### IV.2 – Exploration de cartes

L'exploration de cartes est un des problèmes classiques d'exploration ou de recherche d'intelligence artificielle. Elle est aussi appelée la navigation à travers un territoire comme sur la figure IV.2, qui indique le réseau routier de la Roumanie.



*Figure IV.2 – Exploration de la carte de Roumanie*



### IV.3 – Cube Rubik en 3\*3\*3

La figure IV.3 illustre le jeu du cube rubik 3\*3\*3 et l’animation vidéo d’un robot qui dialogue avec gamin pour résoudre en temps record le jeu tout en décrivant les étapes de résolution par le Robot.



Figure IV.3 – Le jeu du cube rubik 3\*3\*3

### IV.4 – Huit-puzzle ou taquin à huit pièces

Huit-Puzzle (ou taquin à huit pièces) : Le taquin est une sorte de puzzle. On commence avec une grille 3→3 de neuf cases où sont placées huit tuiles étiquetées par les nombres 1 à 8, une des cases restant vide. Une tuile située à côté de la case vide peut être déplacée vers cette case. L’objectif du jeu est d’arriver à obtenir une certaine configuration des tuiles dans la grille (voir la figure IV.4). Le taquin est souvent utilisé pour tester les algorithmes de recherche. En augmentant la taille de la grille, on peut créer les problèmes de plus en plus complexes.

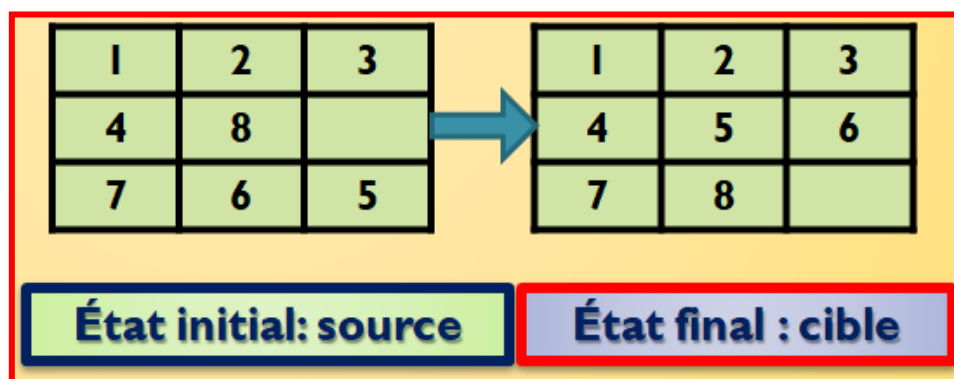


Figure IV.4 – Le jeu du taquin à huit pièces

#### a – Formalisation du problème

La formalisation du problème d’un exemple typique du taquin à 8 pièces est résumé dans le tableau suivant :

État	La description d'un état spécifie l'emplacement des huit pièces et celui de case vide.
État final	N'importe quel état peut être pris comme état initial.
Actions	Dans la formulation la plus simple, les actions sont définies comme des mouvements : gauche, droite, haut ou bas. Les différents sous ensembles d'actions sont possibles selon l'endroit de la case vide.
Modèle de transition	Etant donné un état et une action, cela renvoie l'état résultant en l'échange de position entre le nombre et la case vide.
Test de but	Il vérifie si l'état correspond à la configuration final.
Coût du chemin	Chaque étape coûte 1 et le coût du chemin est donc le nombre d'étapes dans le chemin.

## b - Représentation des états

- Pour le huit-puzzle (ou taquin à huit pièces)
- Tableau de dimension 3\*3 :
  - 5, 6, 7
  - 8, 4, BLANK
  - 3, 1, 2
- Un vecteur de longueur de neuf :
  - 5, 6, 7, 8, 4, BLANC, 3, 1, 2
- Une liste des faits :
  - En haut et à gauche = cinq
  - Moyenne et supérieure = six
  - En haut et à droite = sept
  - Moyen et gauche = huit

Avec le jeu du taquin, on connaît depuis le début quel état on veut, et la difficulté est de trouver une séquence d'actions pour l'atteindre. C'est un problème de planification, où on cherche une suite d'actions pour relier l'état initial à un état final (ou but).

## c - Spécification opérateurs

Il y a souvent plusieurs façons de spécifier les opérateurs, certains seront beaucoup plus faciles à mettre en œuvre etc. Les déplacements possibles sont déplacez vide gauche, avancez à droite du vide, déplacez vide jusqu'à ou descendre le vide. Dans le cas de notre exemple, on a finalement les déplacements suivants :

Déplacez 1 à gauche

Déplacez 1 à droite

- Déplacez 1 place
- Déplacez 1 vers le bas
- Déplacez 2 à gauche
- Déplacez 2 à droite
- Déplacez jusqu'à 2
- Déplacez 2 vers le bas
- Déplacez 3 à gauche
- Déplacez 3 à droite
- Déplacez jusqu'à 3
- Déplacez 3 vers le bas
- Move 4 à gauche
- Etc.

La figure IV.5 illustre les étapes d'évolution de résolution de notre jeu des huit-taquin. Les problèmes de planification sont courants dans la vie de tous les jours. Par exemple, considérons le problème de trouver le chemin le plus court pour atteindre une destination. Les systèmes de planification de voyage qui cherchent des itinéraires en transports en commun, en train ou en avion, le routage dans les réseaux pour l'envoi des paquets et permettre au robot de construire des objets complexes à partir des composants donnés, etc.

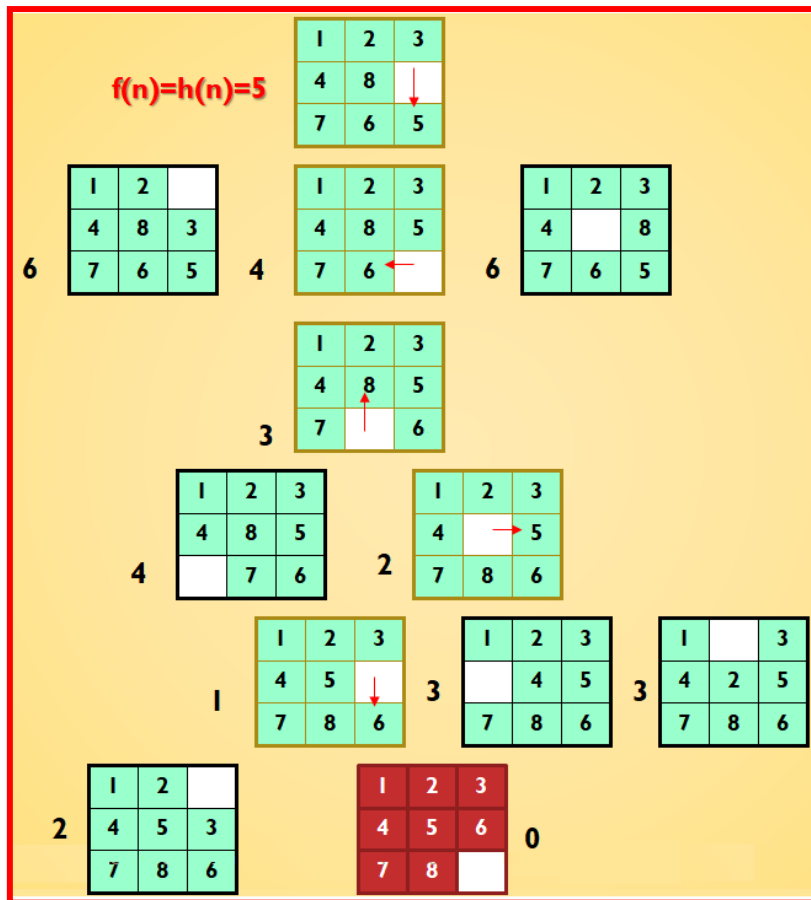
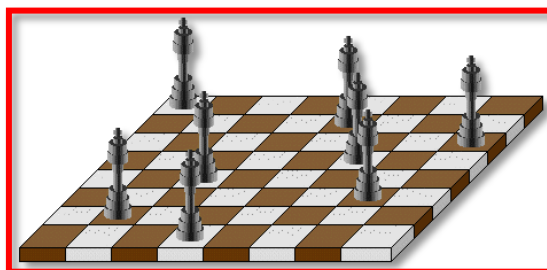


Figure IV.5 – Les étapes de résolutions du problème de huit-taquin

## IV.5 – N-reines

L'objectif du jeu de  $N =$  huit reines est de placer les huit reines sur un échiquier (une grille huit sur huit, tel qu'aucune reine n'attaque une autre reine, c'est-à-dire qu'il n'y a pas deux reines sur la même colonne, la même ligne, ou sur la même diagonale. La configuration donnée dans la figure ci-dessus représente le jeu à huit reines. Pour la suite du cours, on prendra  $N =$  cinq reines à des fins de simplification d'exercice (voir la figure IV.6).



*Figure IV.6 – Le jeu des N-reines*

### a - Formalisation du problème

La formalisation du problème est résumée dans le tableau suivant :

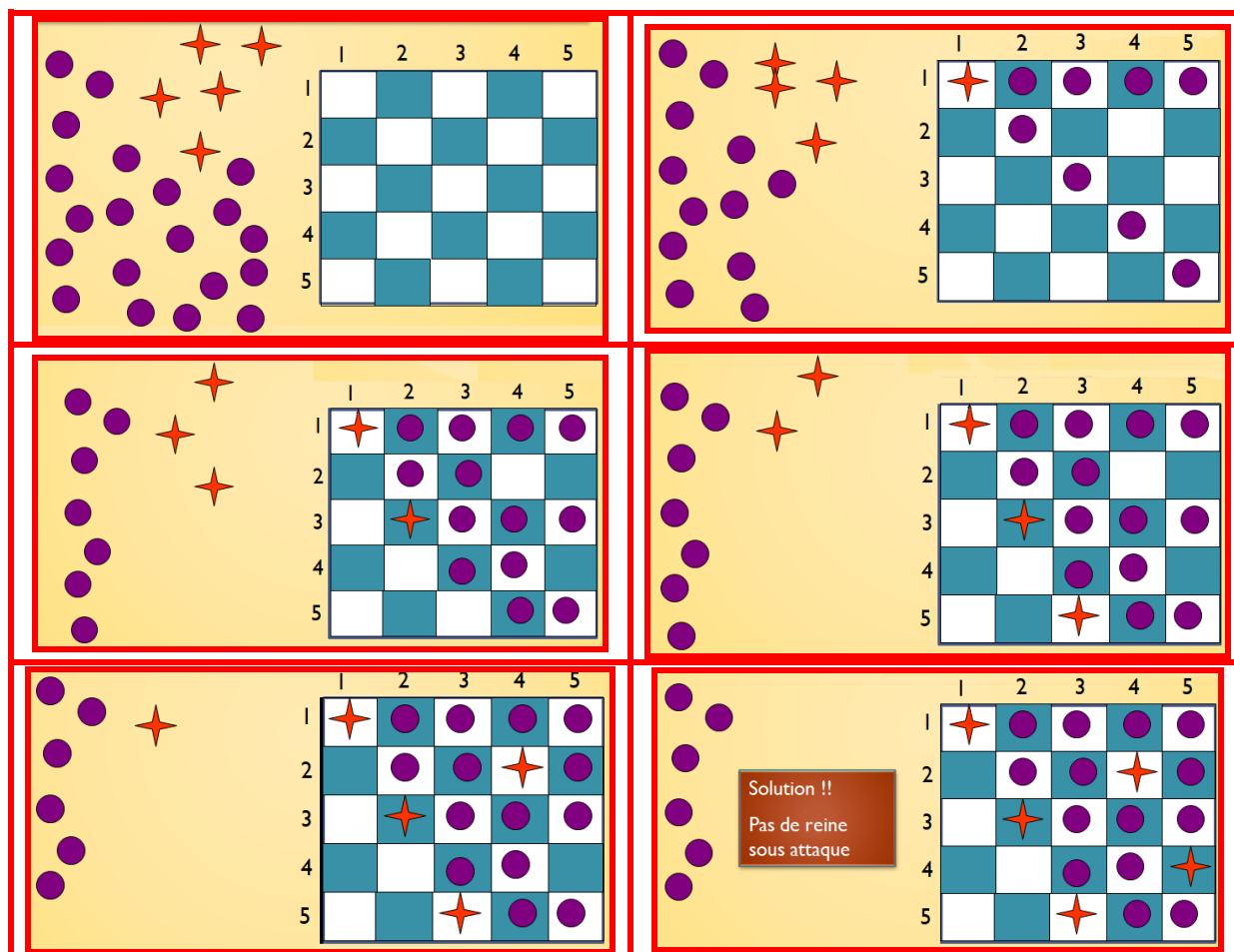
État	Toute disposition de zéro à $N (=$ huit) reines sur l'échiquier constitue un état.
État initial	Aucune reine n'est présente sur l'échiquier.
Actions	Poser une reine sur l'une des cases vides.
Modèle de transition	Retourner l'échiquier avec une reine supplémentaire sur case donnée.
Test de but	Huit reines sont présentes sur l'échiquier et aucune n'est menacée par une autre.

### b - Cinq reines

Avec le jeu des huit reines, on n'est pas intéressé par le chemin mais seulement par l'état but obtenu. C'est un problème de satisfaction de contraintes, qui consiste en un ensemble de variables, des ensembles de valeurs permises pour chaque variable, et un ensemble de contraintes (sur les combinaisons des valeurs des variables). L'objectif est de trouver une évaluation telle que chaque contrainte est satisfaite.

### c– Les étapes de résolution

La figure IV.7 illustre les étapes de résolution du problème des cinq reines.



*Figure IV.7 – Les étapes de résolution du problème des cinq reines*

Les applications Telles que la gestion des pistes d’atterrissage à l’aéroport, la recherche d’un plan de table pour un mariage, la création d’un emploi du temps à l’université, etc. Ce sont tous des problèmes de satisfaction de contraintes (PSC) comme celui des jeux des N reines.

#### IV.6 – Missionnaires et cannibales

Un problème de jeu comme les missionnaires et les cannibales suivant :

- Trois missionnaires et trois cannibales sont sur la rive gauche de la rivière.
- Il y a un canoë qui peut tenir un ou deux personnes.
- Trouver une manière de faire passer chacun à la rive droite de la rivière, sans laisser jamais un groupe de missionnaires dans un endroit dépassé en nombre par des cannibales dans cet endroit.

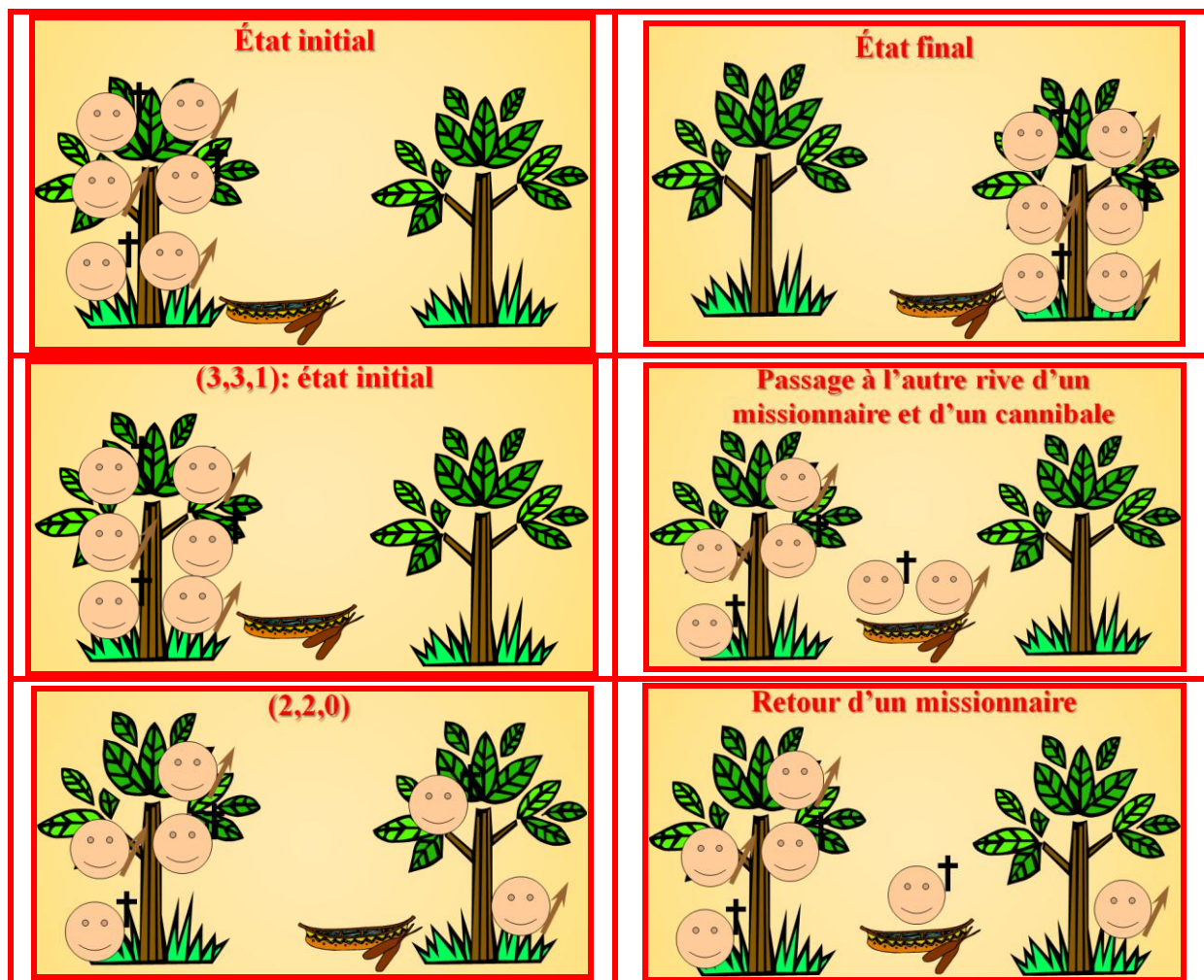
## a - Formalisation du problème

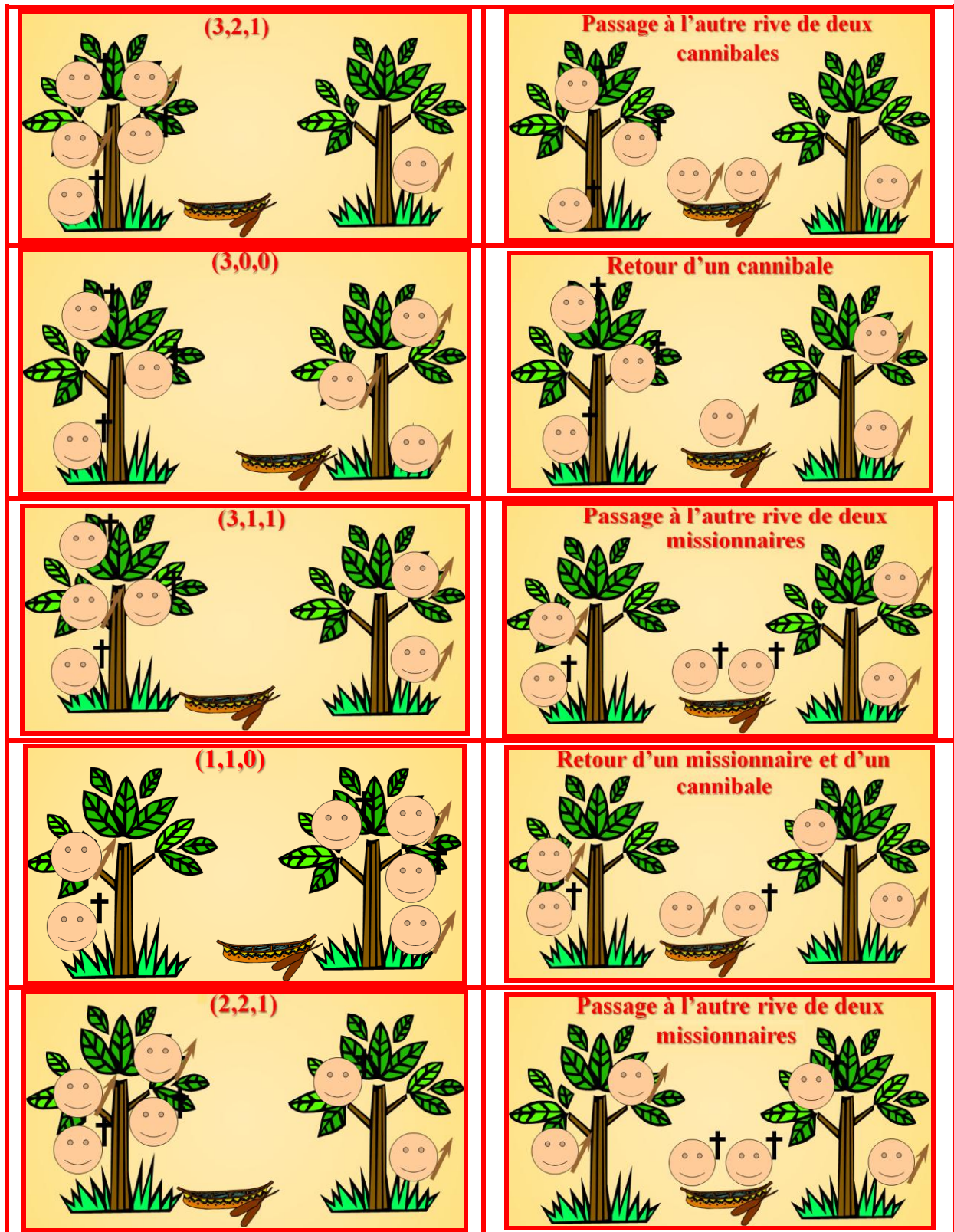
La formalisation du problème est résumée dans le tableau suivant :

État	Trois numéros (i, j, k) représentant le nombre de missionnaires, des cannibales, et des canoës sur la rive gauche de la rivière.
État initial	(3, 3, 1)
Opérateurs	Prendre un missionnaire, un cannibale, deux missionnaires, deux cannibales, un missionnaire et un cannibale à travers le fleuve dans une direction donnée (soit dix opérateurs).
État final	État atteint (0, 0, 0)
Coût du chemin	Le nombre de passages.

## b– Les étapes de résolution

La figure IV.8 illustre les étapes de résolution du problème des missionnaires et des cannibales.





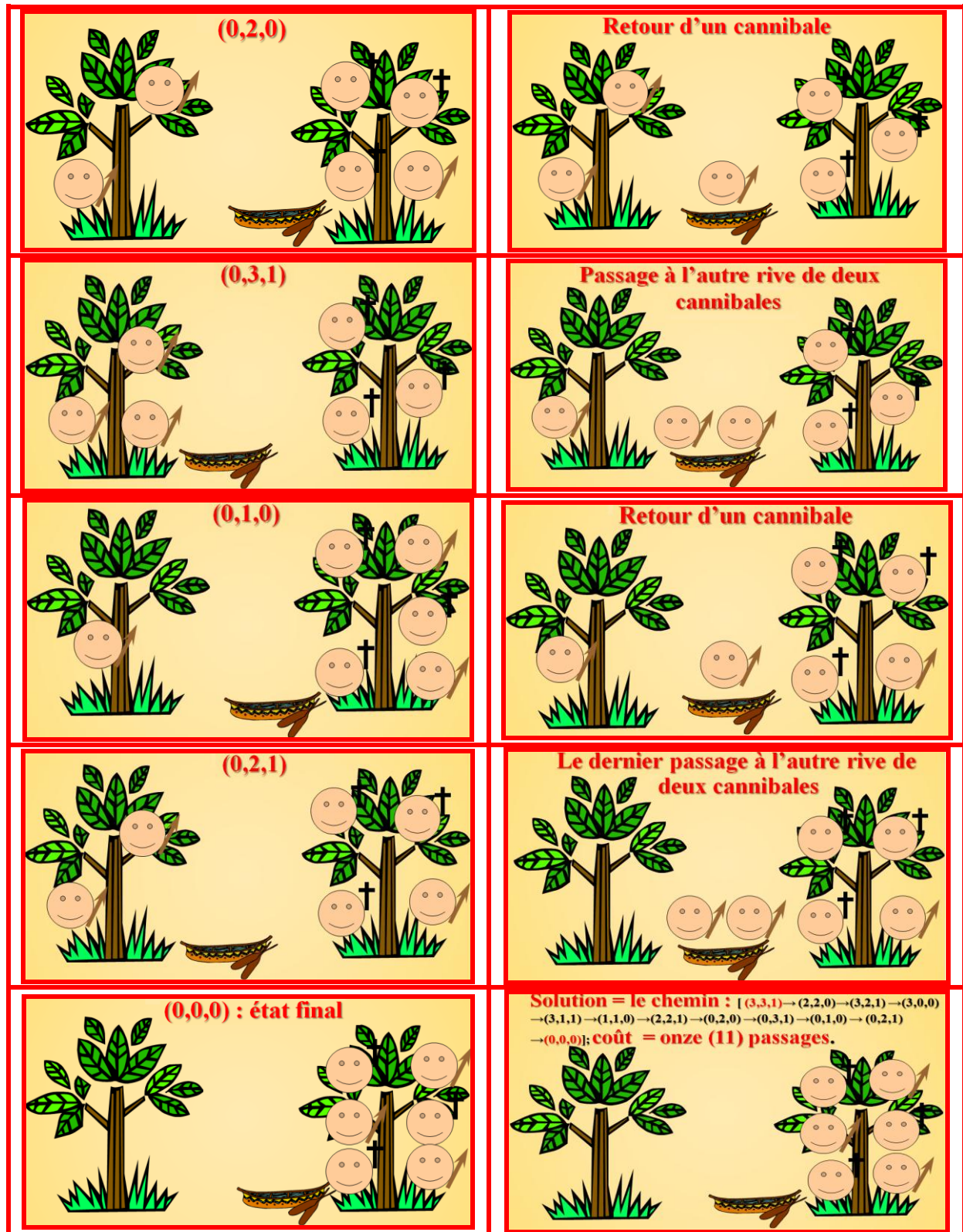
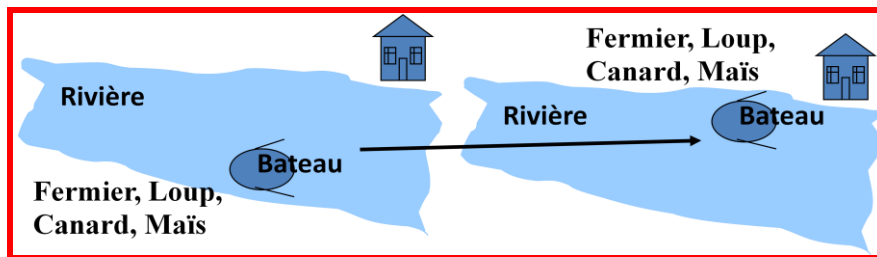


Figure IV.8 – La résolution du problème des missionnaires et des cannibales



## IV.7 – Problème de la rivière

Un fermier souhaite porter le loup, le canard et le maïs à travers la rivière, de sa rive sud vers sa rive nord. Le fermier est le propriétaire d'un petit bateau d'aviron appelé Bounty. Malheureusement le bateau est seulement assez grand pour porter tout au plus le fermier et un autre animal ou plante. Pire encore, s'il laisse seul le loup avec le canard, ce dernier sera mangé par le loup et s'il laisse seul le canard avec le maïs, le canard mangera le maïs (voir la figure IV.9).



*Figure IV.9 – Problème de la rivière*

Le problème revient à ce poser la question : comment le fermier peut-il sans risque transporter le loup, le canard et le maïs au rivage opposé ?

### a - Formalisation du problème

La formalisation du problème est résumée dans le tableau suivant :

La représentation de l'état	L'emplacement du fermier et des objets dans les deux côtés de la rivière [objets sur la Rive-Sud / objets sur la rive nord]: (FLCM / -, FC/LM, M/FLC, etc.).
État initial	Le fermier, le loup, le canard et le maïs sur la rive sud - FLCM/
État final	le fermier, le loup, le canard et le maïs sur la rive nord - /FLCM
Opérateurs	Le fermier prend dans le bateau au plus un élément d'un côté de la rivière à l'autre côté de la rivière (F-Prend-L, F-Prend-C, F-Prend-M, F-Prend-S [lui seul]).
Coût du chemin	Le nombre de passages.

### b - Les stratégies d'exploration

Le problème est résolu en passant de l'état initial à l'état final en appliquant des opérateurs valides dans l'ordre. Ainsi, l'espace des états est l'ensemble des états accessibles à partir d'un état initial particulier comme sur la figure IV.10.



## V - Les algorithmes de base d'exploration

Il existe plusieurs types d'algorithmes de base d'exploration :

- Stratégie d'exploration (sans visibilité ou aveugle) non informée : en largeur, profondeur-premier, profondeur limitée, profondeur itérative et exploration bidirectionnelle.
- Stratégie d'exploration informée (heuristique) : l'exploration est guidée par une fonction d'évaluation : meilleure-première, A\*, IDA\*, SMA\* et exploration faisceau.
- Optimisation dans laquelle l'exploration est de trouver une valeur optimale d'une fonction objective : colline qui s'élève, recuit simulé, algorithmes génétiques, optimisation de colonie de fourmi, optimisation d'essaim de particules.
- Jeu d'un joueur, une recherche de l'adversaire : algorithme de minimax, alpha-bêta élagage.

### V.1 - Quels sont les critères utilisés pour comparer les différentes stratégies d'exploration?

Comme on va considérer différentes stratégies d'exploration de l'espace du problème, on doit examiner quels sont les critères qu'on utilise pour les comparer ? Le tableau suivant illustre les critères utilisés pour comparer les différentes stratégies d'exploration.

Exhaustivité	La stratégie est garantie pour trouver une réponse (si il y en a un).
Optimalité / Admissibilité	Il ne trouve toujours pas une solution à moindre coût? Un algorithme recevable sera de trouver une solution à moindre coût.
Complexité du temps	Combien de temps faut-il pour trouver une solution ?
Complexité de l'espace	Combien de mémoire faut-il pour trouver une solution ?

### V.2 – Les complexités du temps et de l'espace

Les complexités du temps et de l'espace sont mesurés en termes de nombre moyen de nouveaux nœuds qu'on crée lors de l'expansion d'un nouveau nœud est le facteur de ramification qui est  $b$ . Le (maximum) facteur de ramification  $b$  est défini comme les nœuds maximaux créés quand un nouveau nœud est développé. La longueur d'un chemin vers un but est la profondeur  $d$ . La longueur maximale d'un chemin dans l'espace état est  $m$ .

### V.3 – Facteurs de branchement pour certains problèmes

Le huit puzzle ou taquin à huit pièces a un facteur de ramification de deux, treize, donc un arbre d'exploration ou de recherche en profondeur vingt a environ 3,7 millions de nœuds est  $O(b^d)$ . Le cube de Rubik a un facteur de ramification de 13,34. Il y a 901, 083, 404, 981, 813, 616 états différents. La profondeur moyenne d'une solution est d'environ dix huit. Les échecs ont un facteur de branchement d'environ trente cinq, il y a environ  $10^{120}$  États, sachant qu'il y a environ  $10^{79}$  électrons dans l'univers.

### V.4 – Un exemple de jeu : les vacances roumaines

#### a - Formalisation du problème

La formalisation du problème est résumée dans le tableau suivant :

Espace de l'état	Villes en Roumanie
État initial	Ville d'Arad
Objectif	L'aéroport de Bucarest
Opérateurs	Entraînement entre villes
Solution	Séquence de villes
Coût du chemin	Le nombre de villes, la distance, le temps et le carburant.

#### b – Espace des états

La figure V.1 illustre l'espace des états est la carte des routes simplifiée de Roumanie. Si on prend par exemple le problème suivant : on est à Arad en Roumanie et il nous faut atteindre (en parcourant la plus petite distance possible) Bucarest. La figure présente les principales routes de la Roumanie et les distances associées. Soit  $d(A, B)$  la distance routière entre la ville A et la ville B, c'est-à-dire le plus court chemin entre la ville A et la ville B.

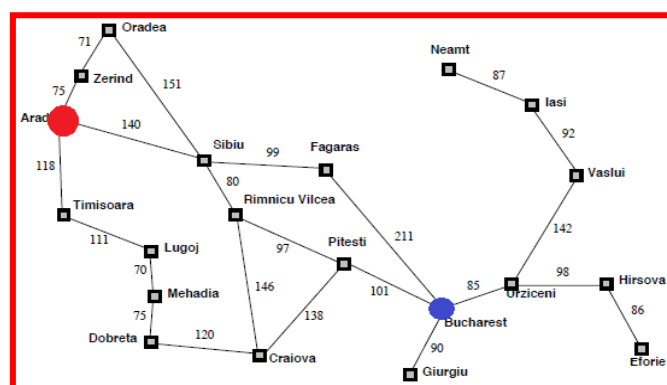


Figure V.1 – Espace des états de la carte des routes de Roumanie

On a par exemple  $d(\text{Zerind}, \text{Sibiu}) = 215$  km. Pour mesurer à quel point on est proche du but lorsqu'on est dans la ville  $X$ , il serait bon de connaître  $d(X, \text{Bucarest})$ , et on aimerait alors prendre  $h(X) = d(X, \text{Bucarest})$ .

### c - Algorithmes génériques d'exploration

L'idée de base consiste en l'exploration hors ligne de l'espace d'états en générant des états successeurs déjà explorés (aussi connu comme les états expansion).

Function GENERAL-SEARCH (*problem, strategy*)

returns a solution or failure

Initialize the search tree using the initial state of problem

loop do

if there are no candidates for expansion, then return failure

Choose a leaf node for expansion according to *strategy*

if node contains goal state then return *solution*

else expand node and add resulting nodes to search tree.

end

### d - Représentation de l'exploration

La figure V.2 illustre la représentation de l'exploration.

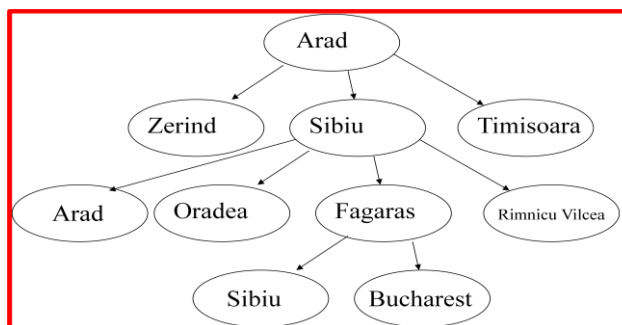
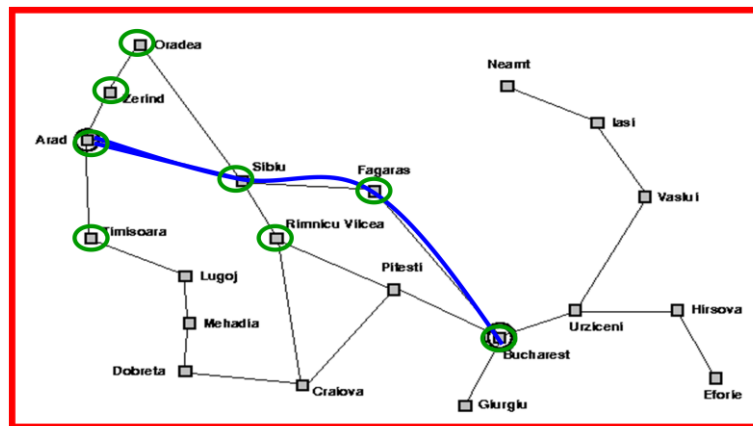


Figure V.2 – La représentation de l'exploration

### e - Solution

La figure V.3 illustre la solution de notre problème de déplacement pour aller d'une ville à une autre.



*Figure V.3 – La solution du problème*

## V.5 - La mise en œuvre du générique de l'algorithme d'exploration

Un bel effet sur cet algorithme d'exploration est qu'on peut utiliser un algorithme unique de faire de nombreux types d'exploration. La seule différence réside dans la façon dont les nœuds sont placés dans la file d'attente. Le choix de la fonction de file d'attente est la caractéristique principale.

```
function general-search(problem, QUEUEING-FUNCTION)
```

```
  nodes = MAKE-QUEUE(MAKE-NODE(problem.INITIAL-STATE))
```

```
  loop do
```

```
    if EMPTY(nodes) then return "failure"
```

```
      node = REMOVE-FRONT(nodes)
```

```
    if problem.GOAL-TEST(node.STATE) succeeds then return solution(node)
```

```
      nodes = QUEUEING-FUNCTION(nodes, EXPAND(node,
        problem.OPERATORS))
```

```
  end
```

## V.6 - Principaux aspects de l'algorithme d'exploration sur l'état de l'espace

Le processus d'exploration pour construire un "arbre d'exploration" commence par la racine, qui est le nœud de départ. Les nœuds feuilles sont des nœuds non expansés (dans la liste des nœuds). Les impasses sont des nœuds qui ne sont pas objectifs et n'ont pas de successeurs, car aucun des opérateurs étaient applicables. Les boucles dans un graphique risquent de provoquer un arbre d'exploration, qui peut être infini, même si l'espace d'état est petit. Il faut changer la définition de la façon dont les nœuds sont ajoutés à la liste conduit à une stratégie d'exploration différente. La solution souhaitée peut être juste l'état final ou un chemin du début à l'état final.

## VI - Stratégies d'exploration non informée (aveugle)

Les stratégies d'exploration non informées dites aussi aveugles utilisent uniquement les informations disponibles dans la définition du problème. Ces stratégies commandent des nœuds sans utiliser aucune information spécifique du domaine. Contrairement aux stratégies d'exploration informées qui pourraient avoir des informations supplémentaires (par exemple, une boussole). Les six stratégies que l'on va étudier sont :

1. Algorithme de parcours en largeur
2. Exploration à coûts uniformes
3. Exploration en profondeur d'abord
4. Exploration en profondeur limitée
5. Exploration itérative en profondeur
6. Exploration bidirectionnelle

### VI.1 Algorithmes d'exploration de base

Les méthodes d'exploration non informée ont accès seulement à la définition de problème. Les algorithmes de base sont résumés dans le tableau suivant :

Algorithmes d'exploration	Développe	Caractéristiques
Exploration en largeur	Les nœuds les moins profonds d'abord	Elle est complète et optimale pour des coûts d'étape unitaires, mais de complexité en espace exponentielle.
Exploration à coût uniforme	Les nœuds ayant le plus bas coût de chemin $g(n)$	Elle est optimale pour des coûts d'étape quelconques.
Exploration en profondeur d'abord	Le nœud non développé le plus profond d'abord	Elle n'est ni complète, ni optimale, mais possède une complexité en espace linéaire.
Exploration limitée en profondeur	Le nœud non développé le plus profond jusqu'à la limite	Elle ajoute une limite de profondeur pour optimiser l'exploration en profondeur.
Exploration en profondeur itérative	Elle appelle l'exploration en profondeur d'abord et augmente les limites de profondeur jusqu'à ce qu'un but soit trouvé.	Elle est complète, optimale pour des coûts d'étape unitaires, possède une complexité en temps comparable à celle de l'exploration en largeur et possède une complexité en espace linéaire.
Exploration Bidirectionnelle	Elle consiste à exécuter deux explorations simultanées : l'une vers l'aval depuis l'état initial et l'autre vers l'amont à partir du but, en espérant qu'elles se rencontrent au milieu.	Elle peut énormément réduire la complexité en temps mais elle n'est pas toujours applicable et peut exiger trop d'espace.

### VI.2 - Comparaison des algorithmes d'exploration

Le tableau suivant nous dresse un tableau de comparaison des algorithmes d'exploration selon les critères du temps, de l'espace, d'optimalité et de complétude.

Critères	Largeur d'abord	Coût uniforme	Profondeur d'abord	Profondeur limitée	Profondeur Itérative	Birectionnelle (si application)
Temps	$b^d$	$b^d$	$b^m$	$b^l$	$b^d$	$b^{d/2}$
Espace	$b^d$	$b^d$	$b^m$	$b^l$	$b^d$	$b^{d/2}$
Optimal?	Yes	Yes	No	No	Yes	Yes
Complètes	Yes	Yes	No	Yes si $l > d$	Yes	Yes

$b$  = facteur de branchement  
 $d$  = la profondeur de la solution  
 $m$  = la profondeur maximale  
 $l$  = la limite de profondeur

## VII - Les algorithmes d'exploration aveugle

### VII.1 - Utilisation des connaissances spécifiques du problème pour aider à l'exploration

Sans l'intégration des connaissances dans l'exploration, on ne peut avoir aucun biais (voir une préférence) sur l'espace d'exploration ou de recherche. Sans un biais, on est forcé de regarder partout pour trouver la réponse. Ainsi, la complexité de l'exploration non informée est intraitable (voir la figure VII.1).

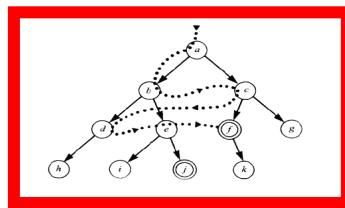


Figure VII.1 – Explorer partout !!!

Avec la connaissance, on peut explorer l'espace d'état, comme si on lui a donné des conseils lors de l'exploration d'un labyrinthe. Les informations heuristique recherche sont des conseils, ce qui conduit à la vitesse spectaculaire en efficacité comme sur la figure VII.2.

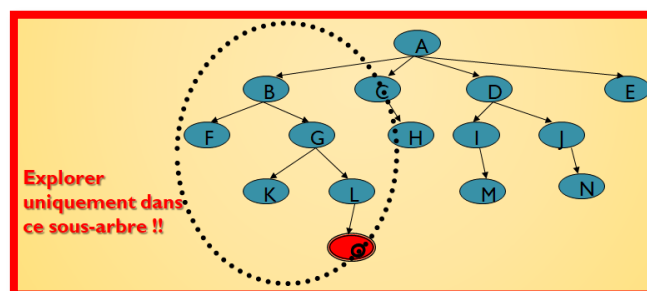


Figure VII.2 – Sous arbre d'exploration



## VII.2 - Plus formellement, lesquelles fonctions heuristiques travaillent?

En tout problème d'exploration où il y a dans la plupart des choix de  $b$  à chaque nœud et une profondeur de  $D$  sur le nœud de but, un algorithme d'exploration naïve devra, dans le pire des cas, l'exploration autour de  $O(b^D)$  nœuds avant de trouver une solution de complexité exponentielle. L'heuristique améliore l'efficacité des algorithmes d'exploration ou de recherche en réduisant le facteur de branchement efficace de  $b$  à idéalement une constante inférieure ou égale à  $b^*$ , telle que :  $1 < b^* \ll b$ .

## VII.3 – Algorithmes d'exploration de base

Les méthodes d'exploration informée peuvent avoir accès à une fonction heuristique  $h(n)$  qui estime le coût d'une solution à partir de  $n$ . Les performances de ces méthodes dépendent de la qualité de la fonction heuristique. Les algorithmes de base sont résumés dans le tableau suivant.

Algorithmes d'exploration	Développe	Caractéristiques
Algorithme générique d'exploration par le meilleur d'abord	Choisir un nœud à développer selon une fonction d'évaluation.	
Exploration gloutonne par le meilleur d'abord	Les nœuds ayant le $h(n)$ minimal.	Elle n'est pas optimal mais souvent efficace.
L'exploration A*	Les nœuds ayant la valeur $f(n)=g(n)+h(n)$ minimal.	A* est complet et optimal, à condition que $h$ soit admissible (pour "Exploration-Arbre") ou consistante (pour "Exploration-Graphe"). La complexité en espace de A* est encore prohibitive.
RBFS (exploration récursive pour le meilleur d'abord)	Il utilise des quantités limitées de mémoire s'il dispose assez de temps. Il peut résoudre les problèmes que A* ne peut pas résoudre parce qu'il manque de mémoire.	Algorithme d'exploration robuste et optimal
SMA* (Exploration A* sous contrainte simplifiée)	Il utilise des quantités limitées de mémoire s'il dispose assez de temps. Il peut résoudre les problèmes que A* ne peut pas résoudre parce qu'il manque de mémoire.	Algorithme d'exploration robuste et optimal

## VII.4 - Fonctions heuristiques

Une fonction heuristique est une fonction  $f(n)$  qui donne une estimation sur le coût d'obtention du nœud  $n$  à l'état but de sorte que le nœud avec le moins de coûts entre tous les choix possibles peut être sélectionné pour le meilleur d'abord. Les trois approches pour définir  $f$  sont :

1.  $f$  mesure la valeur de l'état actuel (sa bonté) :

2.  $f$  mesure le coût estimatif de l'obtention de l'objectif de l'état actuel:  $f(n) = h(n)$  où  $h(n)$  = une estimation du coût pour obtenir de  $n$  à l'état but ;
3. et  $f$  mesure le coût estimé de se rendre à l'état but à partir de l'état actuel et le coût de la voie existante à elle.

Souvent, dans ce cas, nous décomposons  $f$  en :  $f(n) = g(n) + h(n)$  où  $g(n)$  = le coût pour se rendre à  $n$  à partir de l'état initial.

### **a - Approche 1 : les mesures $f$ de la valeur de l'état actuel**

Habituellement le cas lors de la résolution des problèmes d'optimisation, l'exploration d'un état tel que la valeur de la métrique optimisée est  $f$ . Souvent, dans ces cas,  $f$  pourrait être une somme pondérée d'un ensemble de valeurs des composants. Par exemple pour le jeu des N-reines, on a le nombre de reines en attaque. Dans l'exploration de données, on peut par exemple utiliser le prédictive c'est-à-dire l'usage d'une de découverte.

### **b - Approche 2 : Mesures $f$ du coût pour l'objectif**

Un état  $X$  serait mieux qu'un état  $Y$  si le coût estimatif d'obtenir de  $X$  à l'objectif est inférieure à celle de  $Y$ , parce que  $X$  serait plus proche de l'objectif que  $Y$ . Par exemple, pour le jeu du huit Puzzle ou taquin = huit pièces, on peut calculer deux heuristiques  $H_1$  et  $H_2$ , Où  $H_1$  est le nombre de tuiles déplacées (carrés avec le numéro) et  $H_2$  est La somme des distances des tuiles à partir de leurs positions de but.

### **c - Approche 3 : mesures $f$ de coût total du chemin de la solution (fonctions heuristiques admissibles)**

Une fonction heuristique  $f(n) = g(n) + h(n)$  est recevable si  $h(n)$  ne surestime pas le coût pour atteindre l'objectif.

Les heuristiques admissibles sont optimistes, où le coût n'est pas tant que ça important. Cependant,  $g(n)$  est le coût exact pour atteindre le nœud  $n$  à partir de l'état initial. Par conséquent,  $f(n)$  ne surestimer jamais le coût réel pour atteindre l'état final à travers le nœud  $n$ .

### **Théorème**

Une exploration est optimale si  $h(n)$  est recevable. C'est-à-dire : l'exploration en utilisant  $h(n)$  renvoie une solution optimale. Compte tenu  $h_2(n) > h_1(n)$  pour tout  $n$ , il est toujours plus efficace d'utiliser  $h_2(n)$ .  $h_2$  est plus réaliste que  $h_1$  (plus éclairé), si les deux sont optimistes.

## **VIII – Les conclusions**

Frustration avec l'exploration non avertie conduite à l'idée d'utiliser les connaissances d'un domaine spécifique dans une exploration de sorte que l'on peut explorer intelligemment que la partie pertinente de l'espace d'exploration qui a une bonne chance de contenir l'état but. Ces nouvelles techniques sont appelées des stratégies heuristiques d'exploration informée. Même si les heuristiques améliorent la performance des algorithmes d'exploration informée, elles mettent encore beaucoup de temps surtout pour les tailles de grandes instances.

## **IX - Emergence**

C'est une nouvelle façon de procéder à la résolution des problèmes.

### **IX.1 - La résolution des problèmes par émergence : éco-résolution**

Le monde des blocs dont le problème consiste à trouver une séquence d'actions exécutables qui permet d'aller d'une configuration initiale à une configuration finale. Dans la solution proposée, les agents sont les blocs eux-mêmes qui interagissent sur la base de leurs relations : Eco-résolution. Le taquin qui consiste à remettre des palets dans une configuration finale en bougeant un palet à la fois. Dans ce cas, les agents sont les palets qui vont se pousser les uns les autres et engendrer ainsi la séquence de mouvements nécessaires.

### **IX.2 - Huit-puzzle ou Taquin à huit pièces : le principe**

La figure IX.1 illustre le principe du jeu du taquin à huit pièces.

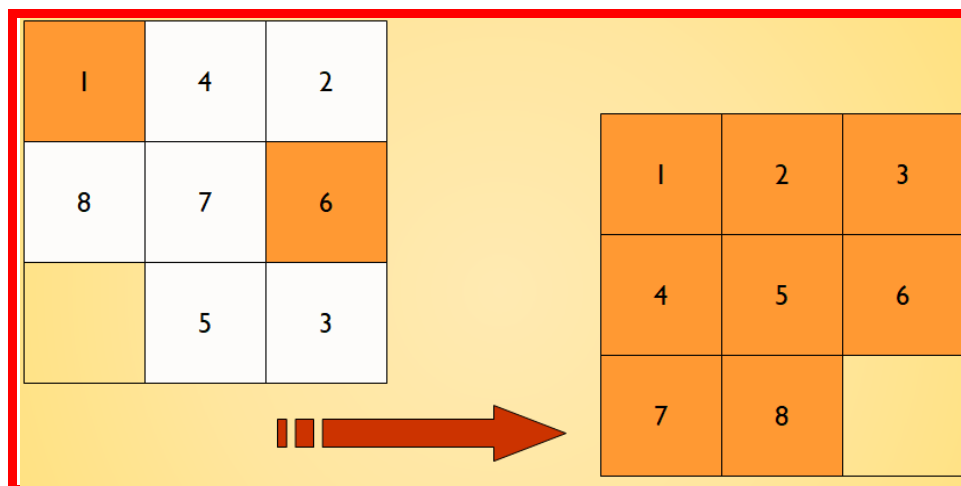


Figure IX.1 – Principe du jeu de taquin à huit pièces

### IX.3 - Taquin & algorithmique

#### Algorithme classique : explorer l'espace des états

Soit  $C$  le chemin (liste d'états) menant d'un état  $E$  à l'état initial. Une fonction heuristique possible est  $h(C) = \text{longueur}(C) + d(E)$ . Soit  $I$  l'état initial.  $F$  une file d'attente de chemins avec priorité  $h$ , et  $D$  un dictionnaire des états déjà visités. On suppose que  $I$  est différent de l'état final.

1. Rentrer le chemin réduit à  $I$  dans  $F$  (avec  $h(I)$ ) et dans  $D$
2. Fini := faux
3. Tant que (non Fini) et ( $F$  non vide) faire
  - a. Sortir un chemin  $C$  de  $F$  de priorité minimale; soit  $E$  le premier élément de  $C$
  - b. Pour tous les successeurs  $S$  de  $E$  et tant que (non Fini) faire
    - i. Si  $S$  est l'état final alors Fini := Vrai et Solution := inverse( $S.C$ )
    - ii. Sinon, si  $S$  n'est pas dans  $D$  alors rentrer  $S.C$  dans  $F$  (avec  $h(S)$ ) et  $S$  dans  $D$

### IX.4 - Éco-résolution : les principes

Chaque éco-agent est défini de la manière suivante :

- Un but représenté par un autre éco-agent avec lequel il entretient une relation de satisfaction ;
- Un état interne. Trois états sont possible : satisfait, en recherche de satisfaction, en recherche de fuite ;

- Des actions élémentaires, qui dépendent du domaine d'application;
- Ce sont des actions de satisfaction ou de fuite;
- Une fonction de perception des agents gêneurs (i.e. les agents qui empêchent l'agent courant d'être satisfait ou de fuir) ;
- Une liste de dépendances (i.e. les agents dont l'agent courant est le but et ces dépendances ne pourront être satisfaits que si l'agent courant se trouve lui-même satisfait).

## IX.5 - Éco-résolution : l'algorithme général

Le comportement élémentaire d'un éco-agent est indépendant du domaine d'application et répond à des principes tels que la volonté d'être satisfait. Un éco-agent cherche à atteindre un état où il est satisfait. S'il ne peut pas y arriver, parce qu'empêché par des gêneurs, il les agresse pour les faire fuir.

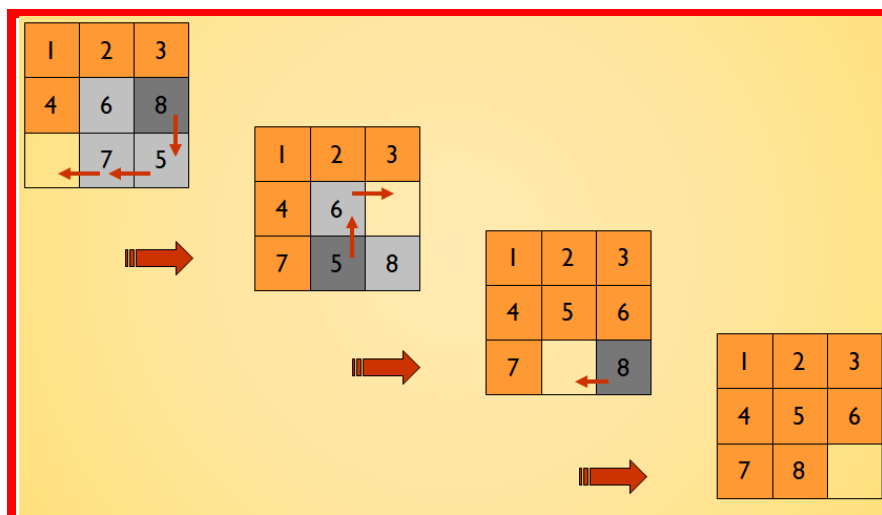
```
procédure trySatisfaction(x : ecoAgent) =  
    si le but de x est satisfait alors  
        pour tous les agents y qui gênent x  
            tryFlee(y, x, but(x))  
dès qu'il n'y a plus de gêneurs doSatisfaction(x)
```

doSatisfaction est dépendant du domaine d'application et se charge d'exécuter l'action dans laquelle l'agent vérifie sa condition de satisfaction. Cette action est possible car les gêneurs sont partis. L'obligation de fuir est lorsqu'un éco-agent est agressé, il doit obligatoirement fuir. Il doit chercher une situation où la situation **c** passé en paramètre de la procédure fuir soit satisfait.

```
// x fuis y avec la contrainte c  
procédure tryFlee(x : ecoAgent, y: ecoAgent, c: ecoAgent)  
si x était satisfait, il devient insatisfait  
soit p = trouverPlacePourFuir(x, y, c)  
si p est nil alors ``pas de fuite possible",  
sinon pour tous les agents z qui gênent x dans sa fuite vers p  
    tryFlee(z, x, p)  
dès qu'il n'y a plus de gêneurs pour fuir, alors doFlee(x, p)
```

## IX.6 - Taquin & éco-résolution

La figure IX.2 illustre l'exécution de l'algorithme éco-résolution sur le jeu du taquin à huit pièces.



*Figure IX.2 – Taquin et éco-résolution*

## X - Développement des systèmes de programmes intelligents

Les deux approches principales pour la résolution des problèmes utilisant la connaissance : l'approche symbolique pour les systèmes experts, qui alimentent le système avec la connaissance et l'approche connexionniste pour les réseaux de neurones, où le système apprend des exemples par lui-même.

### X.1 - Approches symboliques : systèmes à base de connaissances (SBC)

#### a - Les systèmes experts (SE)

Le système expert (ou le système à base de connaissances) est un programme qui encapsule la connaissance d'un certain domaine, normalement obtenu à partir des experts humains de ce domaine. Un système expert peut, dans une certaine mesure, agir comme un substitut aux experts humains dont les connaissances ont été prises. Il s'agit de tenter d'imiter les processus experts de raisonnements et de connaissances pour résoudre des problèmes spécifiques. Les systèmes experts ne remplacent pas les experts, mais ils produisent leurs connaissances et leurs expériences plus largement disponibles, qui permettent aux non experts à mieux travailler. Le système expert agit comme un consultant ou un conseiller. Un système

expert encapsule de la connaissance sous forme de règles et de faits et dispose d'un mécanisme d'inférence lui permettant d'utiliser ces connaissances pour résoudre un problème. Une règle est de la forme "si tel fait est attesté alors effectuer telle action". Une action peut être l'ajout d'un fait; le retrait d'un fait ou la modification d'un fait existant dans la mémoire de travail (voir la figure X.1 et le tableau qui suit)).

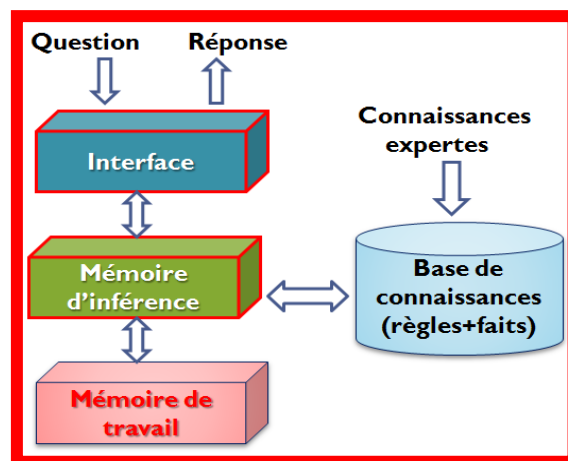


Figure X.1 – Système Expert

## Exemple de règle de DENDRAL

Si le spectre de la molécule présente deux pics  $x_1$  et  $x_2$  tels que  $x_1 - x_2 = M + 28$  et  $x_1 - 28$  est un pic élevé et  $x_2 - 28$  est un pic élevé et au moins l'un des pics  $x_1$  et  $x_2$  est élevé, alors la molécule contient un groupe cétone.

SE	Domaine	Année	Lieu de développement
DENDRAM	Chimie	1969	Université de Stanford - USA
C'est le premier système expert réalisé. Il permet de reconstituer la formule développée d'un composé chimique à partir de sa formule brute et son spectrogramme de masse.			
MYCIN	Médecine	1972-1974	Université de Stanford - USA
C'est un système d'aide à diagnostiquer et soigner les maladies infectieuses du sang et des méningites. Sa réalisation entre 1972 et 1974 a fait intervenir des médecins experts en pharmacologie clinique des infections bactériennes, et des informaticiens experts en IA et informatique médicale. Le résultat est un système expert à base de 200 règles initialement, ensuite 500 règles vers la fin de l'année 1978. C'est un système très performant utilisant une communication quasi-naturelle. Il est basé sur la logique floue, probabiliste, pour la représentation de ses règles dont voici un exemple : "Si le site de la culture est le sang, et à Gram négatif, et l'organisme est de forme bâtonnet, et le patient est un hôte à risque alors il est probable (0.6) que l'organisme soit le pseudomonas aeruginosa".			
EMYCIN	Général	1974	Université de Stanford - USA
Essential MYCIN ou Engine MYCIN ou aussi Empty MYCIN, est le moteur extrait de MYCIN, sans règles qui a été appliqué à plusieurs applications et a servi à la création d'autres systèmes experts.			
OPS	Général	1977	Université de Camegie Mellon - USA
Moteur d'inférence d'un système expert, indépendant de toute application. Utilisé pour l'écriture de la connaissance sous forme de règles de production et permettant de générer de nombreux systèmes experts.			
PROSPECTOR	Géologie	1978	S.R.I. International, Stanford - USA
Système d'aide à évaluer la possibilité d'existence d'un gisement de minerai dans un site étudié. Il compose des			

observations sur les caractéristiques géologiques du terrain considéré et la nature des minéraux trouvés en surface pour prédire l'existence de gisement. Le système utilise la logique floue ou probabiliste avec un raisonnement de type Bayésien et comporte 1600 règles.			
<b>DART</b>	<b>Informatique</b>	<b>1981</b>	<b>IBM - USA</b>
Développé pour le diagnostic des pannes d'ordinateurs et de systèmes. Il comporte 190 règles.			
<b>LPS</b>	<b>Mathématiques</b>	<b>1979</b>	<b>Université Keio - Japon</b>
Système expert de résolution des problèmes de géométrie.			
<b>SOPHIE</b>	<b>Électricité</b>	<b>1975</b>	<b>Université de Stanford - USA</b>
Système d'aide à l'enseignement des techniques de détection des pannes dans un circuit électronique.			
<b>PEGASE</b>	<b>Gestion</b>	<b>1984</b>	<b>École polytechnique de Lausanne - Suisse</b>
Système d'étude des dossiers de demande de crédits.			
<b>CAMA</b>	<b>Automobile</b>	<b>1982</b>	<b>Politecnio, Milan - Italie</b>
Système à 100 règles pour le diagnostic de pannes d'automobiles et le dépannage automatique.			
<b>PDS</b>	<b>Contrôle de processus</b>	<b>1983</b>	<b>Westinghouse - USA</b>
Système de diagnostic en réel d'incident d'exécution de processus automatisés surveillés par ordinateur.			
<b>SONEX</b>	<b>Traitement de la parole</b>	<b>1984</b>	<b>Laboratoire LIMSI, Université Paris 10, Orsay, France</b>
Système d'interprétation de sonagrammes pour la reconnaissance de la parole.			
<b>GREMACOS</b>	<b>Traitement de géomatique</b>	<b>1986-1991</b>	<b>Laboratoire LISI, INSA de Lyon, du PR Kholadi Mohamed-Khireddine</b>
Systèmes d'inférences des connaissances géomatiques.			

## b - Présentation d'une rapide introduction

Les composants du système à base de connaissances (SBC) sont :

1. Base de connaissances (BC) : référentiel des règles et des faits (les productions).
2. Mémoire de travail : (si chaînage avant utilisé).
3. Moteur d'inférence : le système de retenue prévue pour déduire les résultats de l'entrée d'utilisateur et base de connaissances (BC).
4. L'interface utilisateur : les interfaces avec l'utilisateur
5. Contrôle externe + surveillance : accès bases de données externes, de contrôle, etc.

## Pourquoi utiliser les systèmes d'experts?

1. Viabilité commerciale : alors il peut y avoir seulement quelques experts dont le temps est cher et rare, vous pouvez avoir beaucoup de systèmes experts.
2. Systèmes experts peuvent être utilisés n'importe où, à tout moment.
3. Systèmes experts peuvent expliquer leur raisonnement.
4. Commercialement bénéfique : le premier produit commercial de l'intelligence artificielle.



## Faiblesses

1. Systèmes experts sont aussi solides que leur base de connaissances : erreurs dans les règles signifient erreurs dans les diagnostics.
2. Correction d'erreur automatique, l'apprentissage est difficile (bien que la recherche de l'apprentissage de la machine puisse changer).
3. L'extraction des connaissances à partir d'un expert, et codant en forme d'inférences machine est la partie la plus difficile de la mise en œuvre d'un système expert.

## c - Les premiers systèmes experts

Le tableau suivant donne une liste des premiers systèmes experts.

DENDRAL	Utilisé dans la spectroscopie de masse chimique à identifier les constituants chimiques.
MYCIN	Diagnostic médical de la maladie.
Dipmeter	Analyse des données géologiques pour le pétrole.
PROSPECTOR	Analyse des données géologiques pour les minéraux.
XCON /R1	La configuration des systèmes informatiques.
GREMACOS	Inférences des connaissances spatiales des objets géomatiques.

## X.2 - CLIPS (C système de production intégré de langue)

Le langage CLIPS est mis en œuvre en C. Il est librement disponible. Il fournit une langue pour exprimer des règles et utilise chaînage avant de tirer des conclusions. La notation similaire à LISP.

### Exemple de CLIPS

#### Rule

```
(defrule birthday
  (firstname ?r1 Moh)
  (surname ?r1 Ali)
  (haircolor ?r1 Red)
  =>
  (assert (is-boss ?r1)))
```

#### Process

```
(assert (firstname x Moh))
```

```
(assert (surname x Ali))  
(assert (haircolor x Red))  
(run)
```

### **CLIPS> (facts)**

```
f-0 (firstname x Moh)  
f-1 (surname x Ali)  
f-2 (haircolor x Red)  
f-3 (is-boss x
```

## **X.3 - Les langages de programmation de l'intelligence artificielle**

Des langages de manipulation de symbole de haut niveau sont utilisés pour soutenir la mise en œuvre de méthodes d'Amnesty International : LISP pour le traitement des listes et PROLOG pour la programmation logique.

### **a - LISP**

Il est basé sur la notion et les opérations de listes. Tous les problèmes peuvent être décrits sous la forme d'appels de fonction.

#### **Exemple de LISP**

#### **Le plus grand diviseur de deux nombres positifs**

```
(def gcd (a b)  
  (while (<> a b)  
    (if (> a b) (setq a (- a b)) (setq b (- b a)))  
  )  
)
```

#### **Chargement et exécution de fonctions**

```
(load "exemple.ll") or (load "exemple.lsp")  
? (gcd 45 18 )  
= 9
```

### **b -PROLOG**

C'est un Langage déclaratif de haut niveau. Il permet de définir les relations entre différentes entités avec l'aide de la logique. Il procède par des type spécial de la clause (A ←

$B_1 \wedge \dots \wedge B_n$ ) : faits, règles, questions. L'environnement de raisonnement est fait avec un moteur d'inférence intégré. La réponse à une question est faite à l'aide d'un raisonnement logique. Le raisonnement (en arrière) est guidée par objectif ou but.

## Exemples de Prolog

Domain

e = integer

l = e\*

predicates

minimum(e,l)

append(l,l,l)

sort(l,l)

clauses

minimum(\_,[\_]).

minimum(A,[B|R]) :- A<B, minimum(A,R).

append([\_],L,L).

append([A|L1],L2,[A|L3]) :- append(L1,L2,L3).

sort([\_],[\_]).

sort([A|Z],L) :- minimum(A,Z),sort(Z,L1), L=[A|L1].

sort([B|Z],L) :- not(minimum(B,Z)),

append(Z,[B],L1),sort(L1,L).

But

sort([6 4 8 2], L)

True

L = [2 4 6 8]

DOMAINS

s = SYMBOL

PREDICATES

father(s, s)

son(s, s)

grandfather(s, s)

CLAUSES

```
father(ali, salim).
father(salim, farid).
son(X,Y) :- father(Y, X).
grandfather(X, Y) :- father(X, Z), father(Z, Y).
```

## But

```
write(" hello ... "), nl,
grandfather(X, farid),
write(Y), nl, nl.
> hello
X = ali
1 solution
```

## c - Comparaison des manipulations symboliques et les techniques traditionnelles

Le tableau suivant résume la comparaison entre les deux langages LISP et PROLOG.

Langages de programme classique	LISP	PROLOG
Calculs numériques	Manipulation symbolique	Manipulation symbolique
Le langage de principe de Neumann consiste en une séquence de commandes exécutées dans un ordre prédéfini	Approche fonctionnelle séquence d'évaluation de la fonction-expressions ( $\lambda$ -calcul)	Approche de la relation sur la base de la logique mathématique (calcul des prédicats)
Éléments principaux: commandes	Éléments principaux: fonctions (procédures)	Éléments principaux: prédicats (relations entre objets)
Procédural (exécution dans un ordre prédéfini)	Procédural	Déclaratif (la seule définition de la description du problème)
Le mécanisme d'exécution doivent être défini par le programmeur	Le mécanisme d'exécution doivent être défini par le programmeur	Intégré dans le mécanisme d'exécution (raisonnement par objectifs, faire marche arrière dans la stratégie de recherche)
La structure du programme et des données est différente	La structure de programme et des données est la même (peut produire, exécuter d'autres programmes, peuvent se modifier)	La structure de programme et des données est la même (peut produire, exécuter d'autres programmes, peuvent se modifier)
Bonne lisibilité	Dure à lire	Facile à lire

## XI - Approche connexionniste

### XI.1 - Apprentissage machine : réseaux de neurones

Les réseaux de neurones peuvent être utilisés pour répondre aux questions suivantes :

1. La reconnaissance des formes : Est-ce que l'image contient un visage?
2. Les problèmes de classification : Est-ce cellule défectueuse?
3. Prédiction : Compte tenu de ces symptômes, le patient a la maladie X.

4. Prédiction : Le comportement de la prédiction boursière.
5. Écriture : Est-ce que le caractère est reconnu?
6. Optimisation : Trouver le plus court chemin.

Les réseaux de neurones artificiels sont une tentative de bas en haut pour modéliser le fonctionnement du cerveau. Deux grands domaines d'activité sont le biologique, qui tente d'essayer de modéliser des systèmes de neurones biologiques, et le computationnel, où les réseaux neuronaux artificiels sont biologiquement inspirée mais pas nécessairement biologiquement plausible. Alors, on peut utiliser d'autres termes, tels que le connexionnisme, le traitement distribué parallèle ou l'adaptive la théorie des systèmes. Les intérêts dans les réseaux de neurones diffèrent selon la profession.

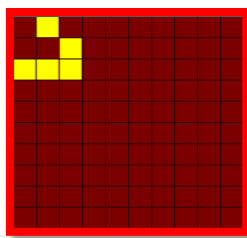
## **XII -Nouvelle Intelligence Artificielle (l'intelligence computationnelle ou les systèmes complexes)**

### **XII.1 - Vie artificielle et systèmes complexes**

La vie artificielle est une tentative de mieux comprendre la vie réelle par la modélisation de facto des entités que l'on connaît. En ce qui concerne les motivations, la vie artificielle aurait été surnommé une autre approche pour étudier la vie intelligente, n'eut été pour les propriétés émergentes dans la vie qui motive les scientifiques à explorer la possibilité de créer artificiellement la vie et s'attendre à l'inattendu. Une propriété émergente est créée quand quelque chose devient plus que la somme de ses parties.

### **XII.2 - Vie artificielle et automates cellulaires**

Les automates cellulaires (CA) est un tableau de cellules à N dimensions qui interagissent avec leurs cellules voisines selon un ensemble prédéterminé de règles, de générer des actions, ce qui peut à son tour déclencher une nouvelle série de réactions sur lui-même ou ses voisins. L'exemple le plus connu est la vie de Conway, qui est un 2-état 2-D CA avec des règles simples (voir à droite) appliquée à toutes les cellules simultanément pour créer des générations de cellules à partir d'un motif initial comme sur la figure XII.1.

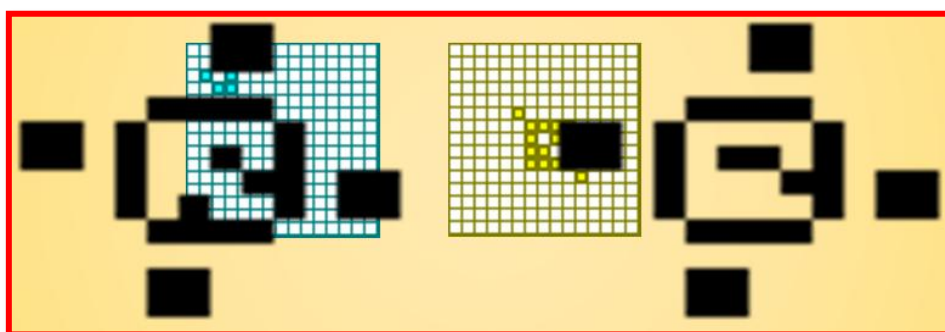


*Figure XII.1 – La vie de Conway*

<b>Conway's Life: Rules</b>
Une cellule vivante avec 0-1 8-voisins meurt d'isolement
Une cellule vivante avec 4+ 8-voisins meurt de surpopulation
Une cellule morte avec 3 voisins vivant devient une cellule vivante
Toutes les autres cellules ne sont pas affectés

### **XII.3 - Automates cellulaires - le jeu de la vie**

La figure XII.2 illustre les automates cellulaires. Les règles de transition simples donnent lieu à des modèles complexes (structures émergentes).



*Figure XII.2 – Automates cellulaires*

## **XIII - Nouveaux algorithmes de tri basé sur le calcul naturel**

### **XIII.1 - Algorithme traditionnel de tri**

Class vector ...

```
import java.util.*;
```

```
public class Vector {
```

```
int [] A;
```

```
Scanner input = new Scanner(System.in);
```

```
public Vector(int nbe) {
```

```
A = new int [nbe];
```

```
System.out.println("enter data ");
```

```
for (int i=0; i<nbe; i++) {
```

```
System.out.print(" enter element : "); A[i] = input.nextInt(); }
```

```
}
```

```
Class vector ...
```

```
public void bubblesort() {
```

```
int temp;
```

```
for (int i=A.length-1; i>0; i--)
```

```
for (int j=0; j<i; j++)
```

```
if (A[j] > A[j+1]) {
```

```
temp = A[j];
```

```
A[j] = A[j+1];
```

```
A[j+1] = temp;
```

```
}
```

```
}
```

```
public void insertionsort() {
```

```
int value,i,j; boolean done;
```

```
for (i=1; i<=(A.length-1); i++) {
```

```
value = A[i]; j = i-1; done = false;
```

```
do
```

```
if (A[j] > value) {
```

```
A[j+1] = A[j--];
```

```
if (j<0)
```

```
done = true;
```

```
}
```

```
else
```

```
done = true;
```

```
while (!done);
```

```
A[j+1] = value;
```

```
}
```

```
}
```

Class vector

```
public void displayvector() {
    System.out.println(" vector display ");

    for (int i=0; i< A.length; i++)
        System.out.println(A[i]);

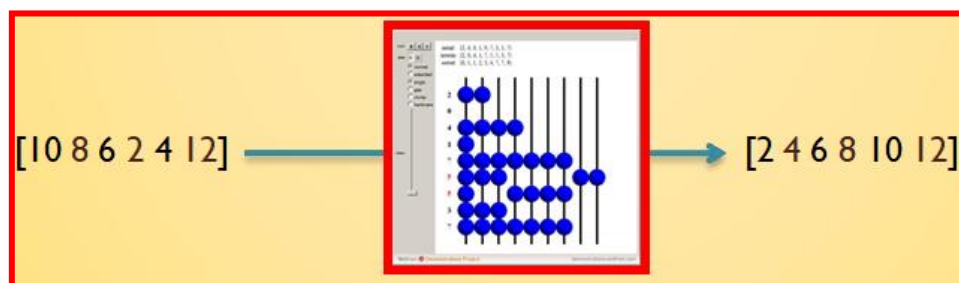
}

public void acceptancetest() throws BugSort {

    for (int i=0; i<A.length-1; i++)
        if (A[i] > A[i+1]){
            System.out.println(" Warning Wrong Sorting !!!!");
            throw new BugSort(); // raise exception ...
        }
    }
} // end of class Vector
```

### XIII.2 - Tri des boules : Une nouvelle façon pour le tri sur la base de l'informatique naturelle

Le tri des boules est une méthode de commande d'un ensemble de nombres entiers positifs en imitant le processus naturel de chute d'objets au sol. Le nombre de boules sur chaque ligne horizontale représente l'un des numéros de l'ensemble à être triés, et il est clair que l'état final représentera l'ensemble trié (voir la figure XIII.1).



*Figure XIII.1 – Tri des boules*



### XIII.3 - Boules-Tri étendu

Le mode étendu (anti-gravité) permet l'inclusion de tous les entiers, avec des boules négatives hautes tandis que les boules positives basses comme sur la figure XIII.2.

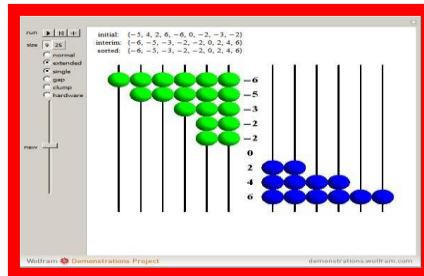


Figure XIII.2 – Boules-tri étendu

### XIII.4 - Boules-tri-AC mise en œuvre

La figure XIII.3 illustre le processus naturel de chute d'objets au sol. Une démonstration sur Netlogo est présentée aux étudiants sur la figure XIII.4.

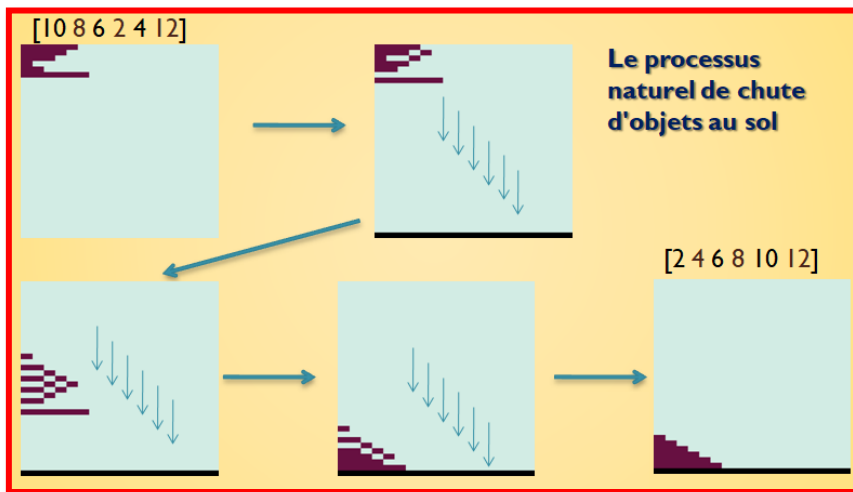


Figure XIII.3 – Le processus naturel de chute d'objets au sol



Figure XIII.4 – Démo Netlogo

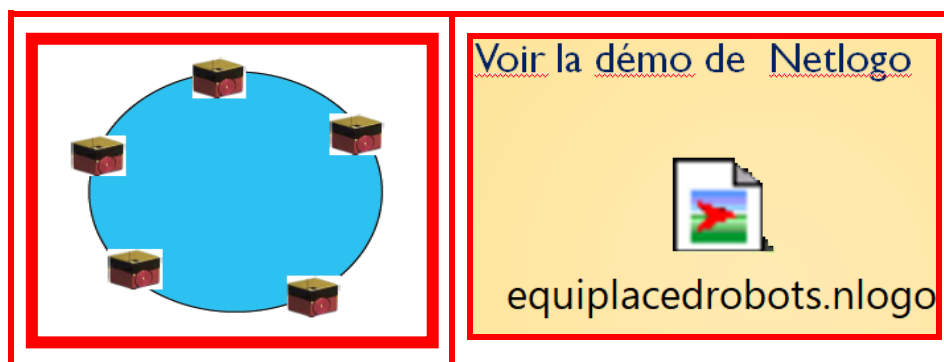
Il faut procéder à l'entrée des données dans la démo Netlogo sous la forme suivante :

[10 -5 6 4 -2 11 -8 3]

## XIV - L'intelligence collective

### XIV.1 - Comment placer n agents en cercle de manière équirépartie ?

La figure XIV.1 illustre la manière de placer n agents de manière équidistante d'un point.



*Figure XIV.1 – Comment placer n agents en cercle de manière équirépartie*

## XV – Les conclusions

Pour conclure, l'intelligence artificielle est un domaine très fascinant. Il peut nous aider à résoudre des problèmes difficiles du monde réel, créant de nouvelles opportunités dans les affaires, l'ingénierie et de nombreux autres domaines d'application. Même si la technologie de l'Intelligence artificielle est intégrée dans le tissu de la vie quotidienne. Les promesses ultimes de l'intelligence artificielle sont encore loin des décennies et les progrès nécessaires dans les connaissances et la technologie exigeront un effort soutenu de recherche fondamentale.

## XVI – Les références

1. Intelligence artificielle avec plus de 500 exercices, S. Russell et P. Norvig, 1200 pages, Edition Pierson, 3<sup>ème</sup> édition, 2012.
2. Introduction to Expert Systems, Jackson
- I. Bratko, "Prolog Programming for Artificial Intelligence", Prentice Hall, 2000.
3. Expert System Principles and Programming, Giarrantano and Riley
4. C. Queinnec, "Langage d'un autre type : LISP", Eyrolles, 1983

5. <http://www.ghg.net/clips/CLIPS.html>
6. AI communications, The European Journal on Artificial Intelligence, IOS Press, The Netherlands, published quarterly.
7. AI Magazine, AAAI Press, USA, published quarterly.
8. Artificial Intelligence, Elsevier Science, The Netherlands, published monthly.
9. Artificial Life, MIT Press, Cambridge, Massachussets, published quarterly.
10. IEEE Intelligent Systems, IEEE Press, publié 6 fois par an.
11. JETAI : Journal of Experimental and theoretical Artificial Intelligence, Taylor and Francis Ltd, UK., published quarterly.
12. Journal of the ACM, ACM, NY, publié 6 fois par an.

## XVII – Le cas n°1: Système complexe SIG pour le suivi de la remontée des eaux de la wilaya d’El-Oued Souf

Notre objectif est de développer un outil d’aide à la prise de décisions et à la simulation du phénomène qui peut être sous la forme d’un système complexe du type SIG pour le suivi de la remontée des eaux de la wilaya d’El Oued Souf comme sur la figure XVII.1.

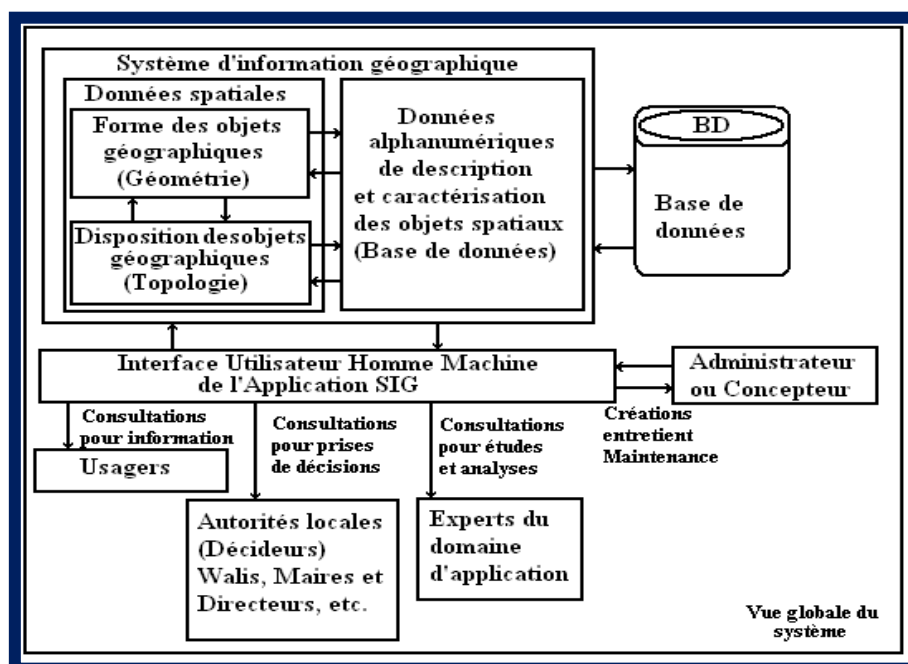
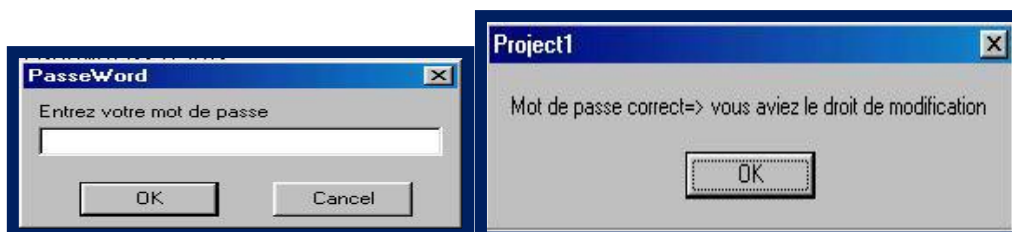


Figure XVII.1 – Vue globale du système

### XVII.1 – La présentation de l’interface utilisateur

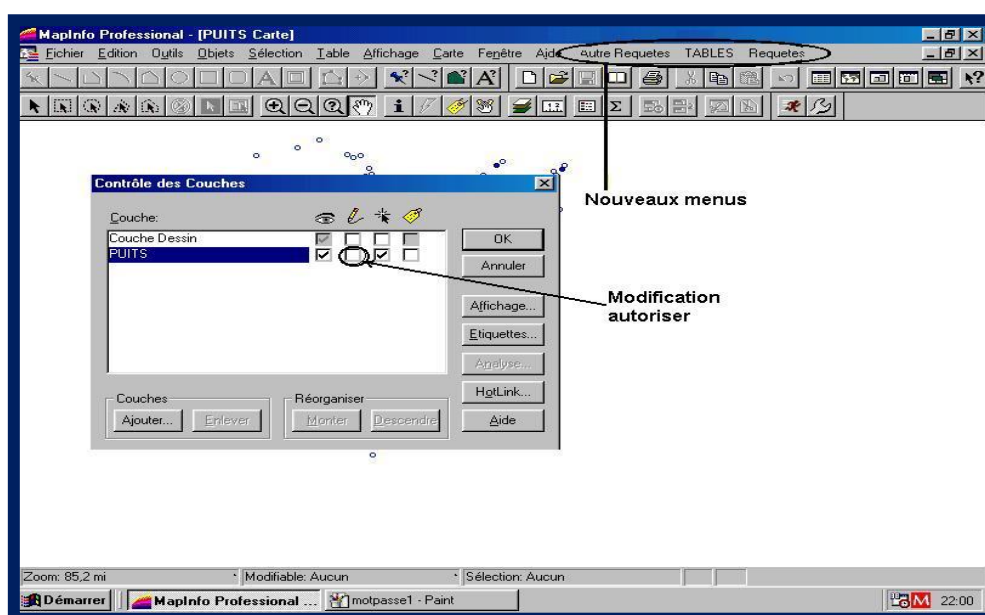
La réalisation de l’interface a été faite à l’aide de Delphi 6 ( environnement de programmation ayant comme langage le Pascal objet). On l’a utilisé comme outil d’interface

car le MapBasic n'offre pas cette option. Ce logiciel possède des concepts des langages orientés objets : l'héritage, la modularité, la surcharge, etc. Après la page d'accueil on trouve la fenêtre suivante qui nous demande de faire entrer le mot de passe. Si le mot de passe est correct, alors on est en mode administrateur, pour les concepteurs du système et l'administrateur. Dans ce cas, on se trouve devant les pages suivantes de la figure XVII.2 :



*Figure XVII.2 – Mot de passe d'accès au projet*

Si le mot de passe est incorrect, on accède au mode consultation, pour les usagers, les décideurs et les experts, on accède à l'interface de travail de la figure XVII.3.



*Figure XVII.3 – Accès à l'interface de travail*

## **XVII.2 - La couche "Population"**

Elle nous donne toutes les statistiques sur la population et les logements de chaque commune de la wilaya d'El Oued Souf comme sur la figure XVII.4 et le tableau suivant.

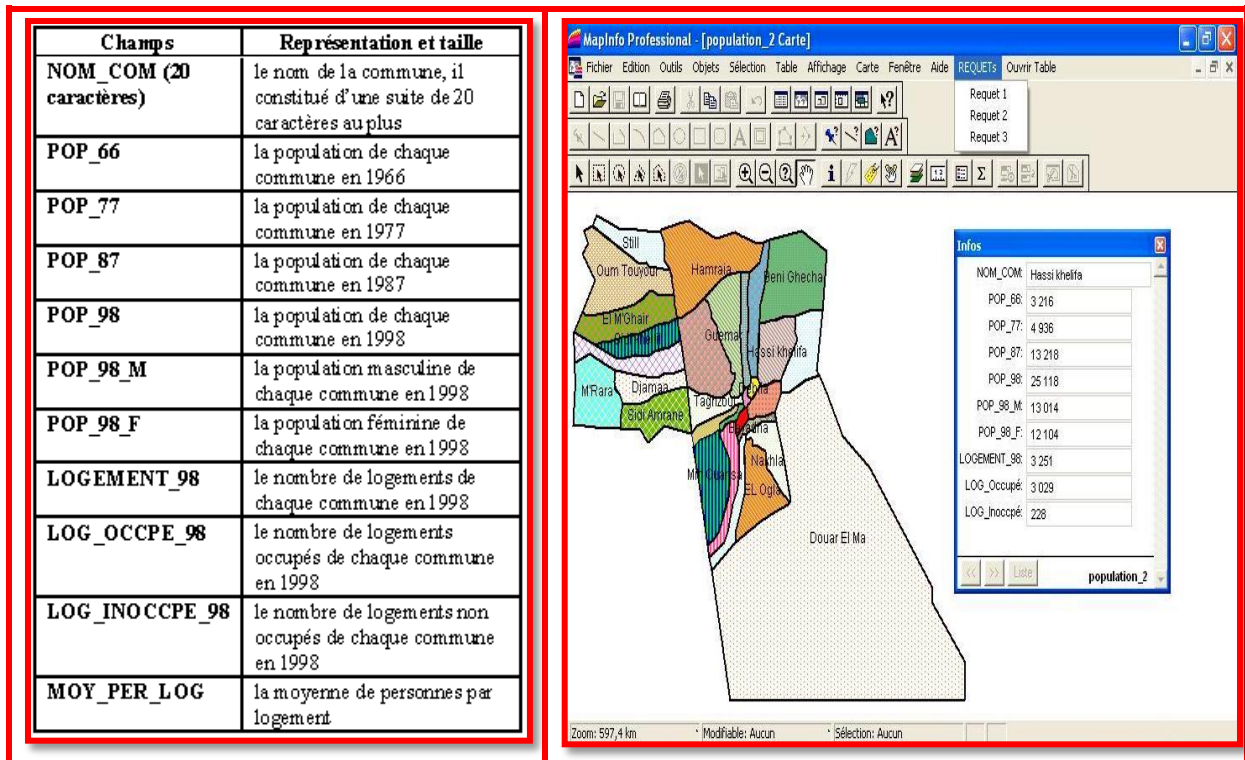


Figure XVII.4 – La couche population

### XVII.3 - La couche "Puits"

La couche "PUITS" illustre les puits avec quelques données comme la salinité de chaque commune de la wilaya d'El Oued Souf comme sur la figure XVII.5 et les tableaux suivants.

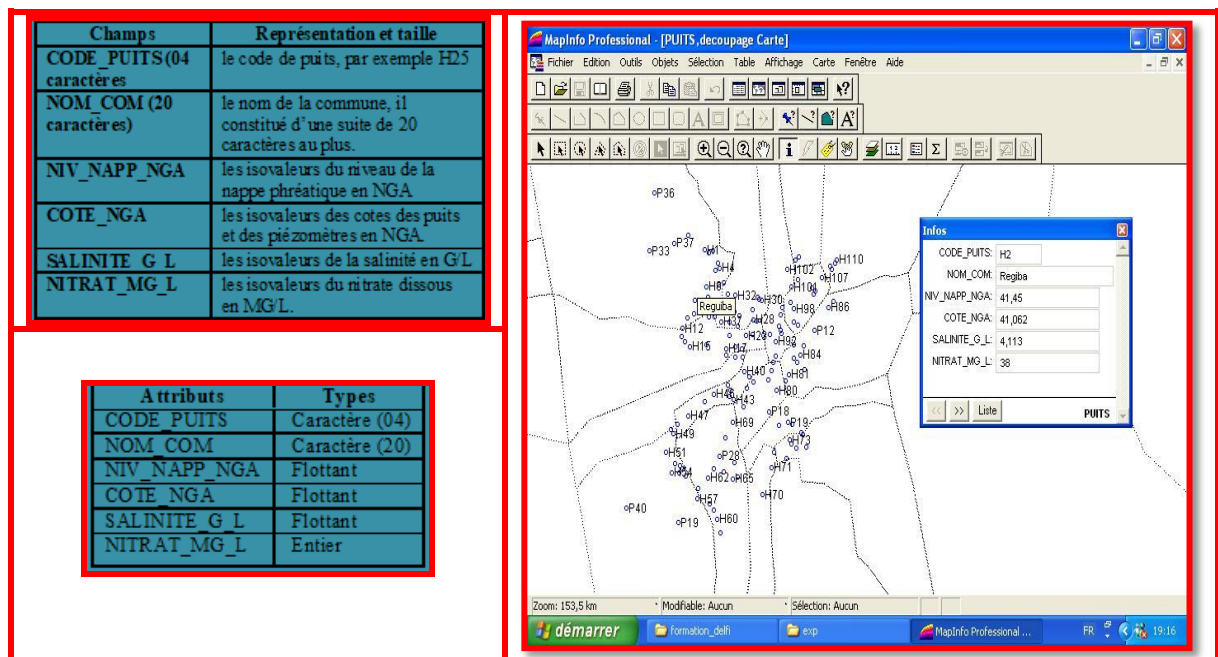
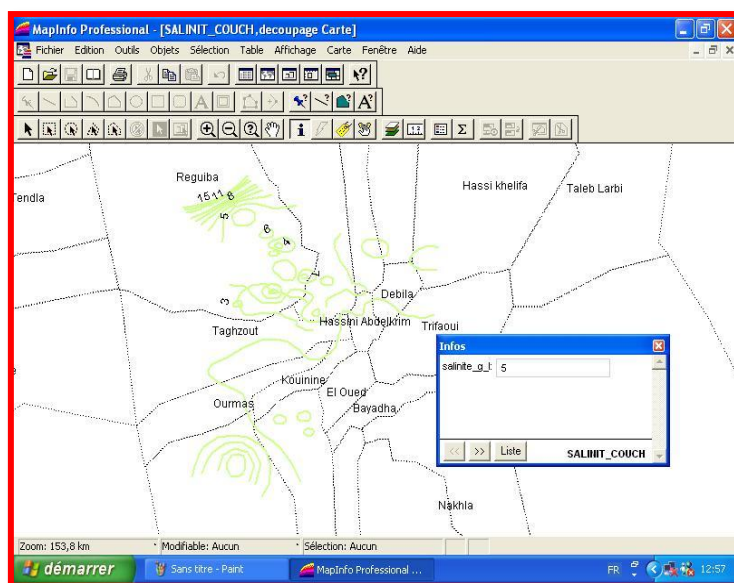


Figure XVII.5 – La couche puits

## XVII.4 - La couche "Salinité"

La couche "SALINITE" illustre les lignes isovaleurs de la salinité comme sur la figure XVII.6.

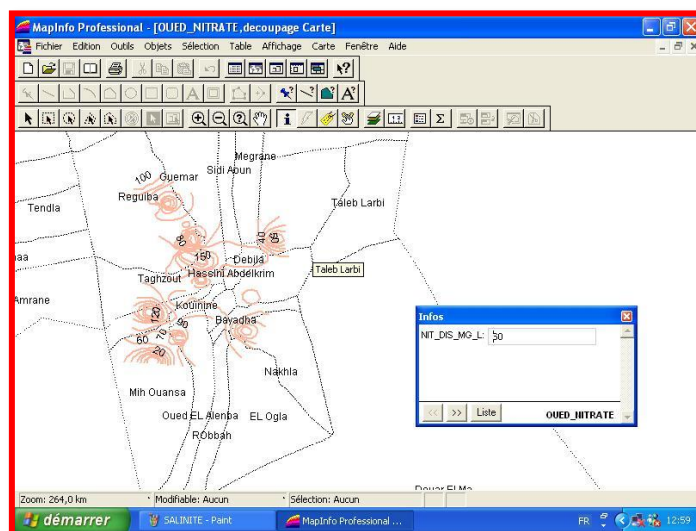


Attributs	Types
Salinite_g_1	Entier

Figure XII.6 – La couche salinité

## XVII.5 - La couche "Nitrate dissous"

La couche "NITRATE\_DISSOUS" illustre les lignes isovaleurs du nitrate dissous comme sur la figure XVII.7.

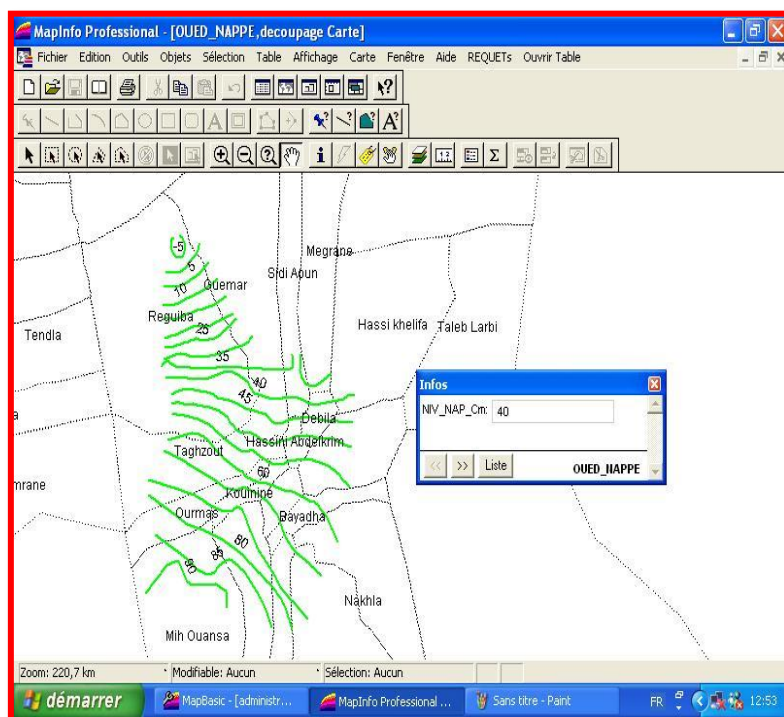


Attributs	Types
NIT DIS MG L	Entier

Figure XVII.7 – La couche de nitrate dissous

## XVII.6 - La couche "Nappe phréatique"

La couche "Nappe phréatique" illustre dans la figure XVII.8.



Attributs	Types
NIV_NAP_NGA	Entier

Figure XVII.8 – La couche de nappe phréatique

## XVII.7 – Les requêtes

On a réalisé 12 requêtes. On va choisir un ou deux exemples pour montrer les possibilités de notre système complexe. On peut aussi faire appel à des requêtes spatiales telles que le pointé et le zonage avec la résolution des problèmes des optimisations par l'indexation spatiale.

### XVII.7.1 – La requête "Couche de population"

La requête suivante va donner comme résultat la mise en valeur des communes qui ont une population supérieure à 20 000 habitants en 1998 (voir la figure XVII.9).

```
SELECT NOM_COM
FROM POPULATION
WHERE POPULATION.POP_98 > 20000
```

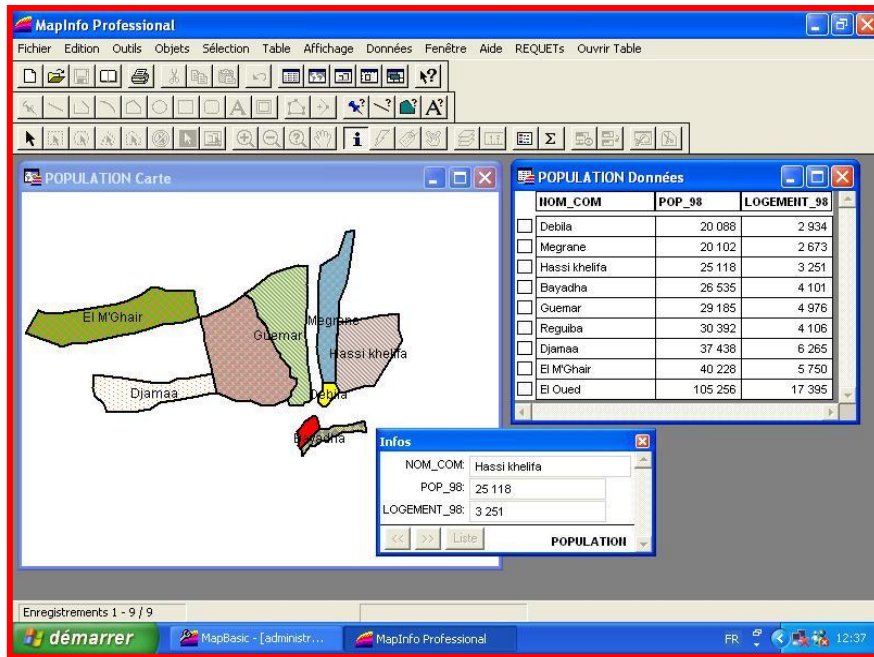


Figure XVII.9 – Requête couche population

### XVII.7.2 – La requête "Couche de puits"

La requête ci-après va donner comme résultat la mise en valeur les puits qui sont en danger à cause de la salinité (< 4g/l) voir la figure XVII.10.

```
SELECT CODE_PUITS
FROM PUITES
WHERE SALINITE < 4
```

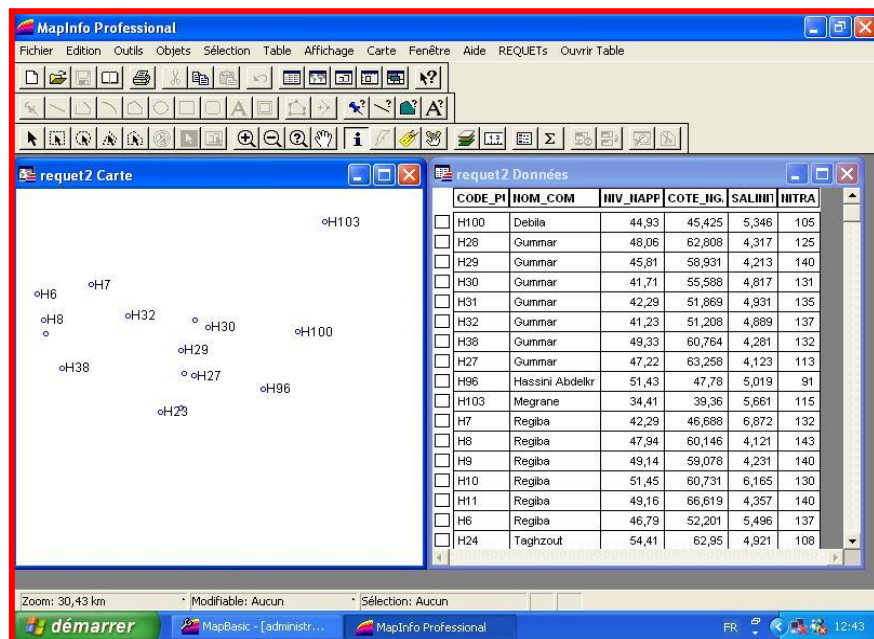


Figure XVII.10 – Requête couche de puits



## XVII.7.3 – La représentations graphiques

MapInfo nous offre aussi les possibilités d'effectuer des statistiques avec différents types de représentations : 3D, AIRES, BARRES, HISTOGRAMMES, SECTEURS, etc. On va citer deux exemples sur la figure XVII.11.

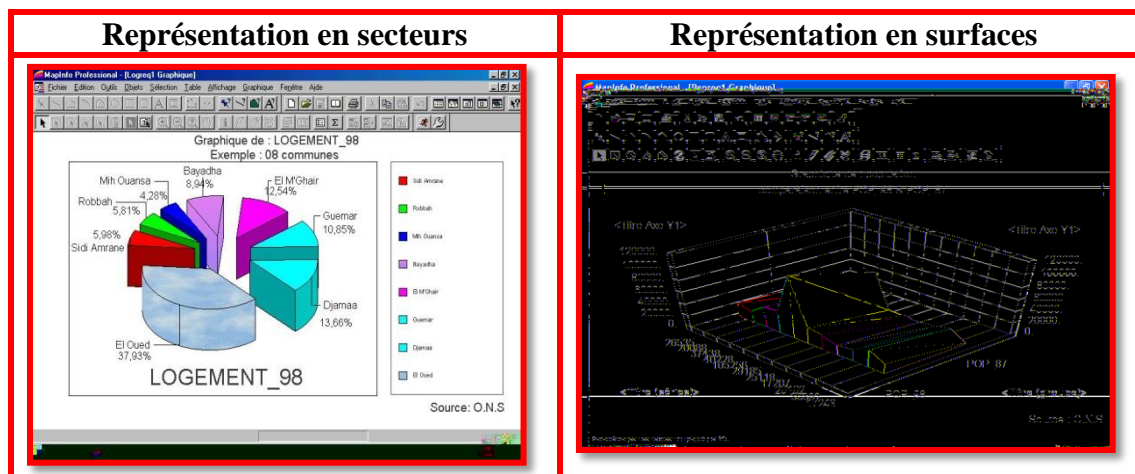


Figure XVII.11 – Représentation graphique

## XVIII – Le cas n°2 : Système complexe SIG pour l'étude de l'évolution de la répartition de la population de la wilaya d'Adrar

Le recours à un système complexe SIG répond à l'objectif d'avoir une base de données (voir de connaissances) géographiques à la disposition des utilisateurs. Puis, il s'agit surtout de développer des outils de manipulations et d'analyses pour répondre aux interrogations des différents types d'utilisateurs. Notre objectif est de mettre à leur disposition des outils informatiques qui permettent de répondre à leurs préoccupations d'une manière intelligente, du genre :

1. Qu'y a-t-il en ce lieu ?
2. Quels sont les types de population qui occupent un lieu ?
3. Est-il opportun de construire équipement scolaire ou sportif en ce lieu ?
4. Etc.

### XVIII.1 - Le modèle conceptuel

La figure XVIII.1 illustre le modèle conceptuel de notre application via le modèle entité association de Merise.

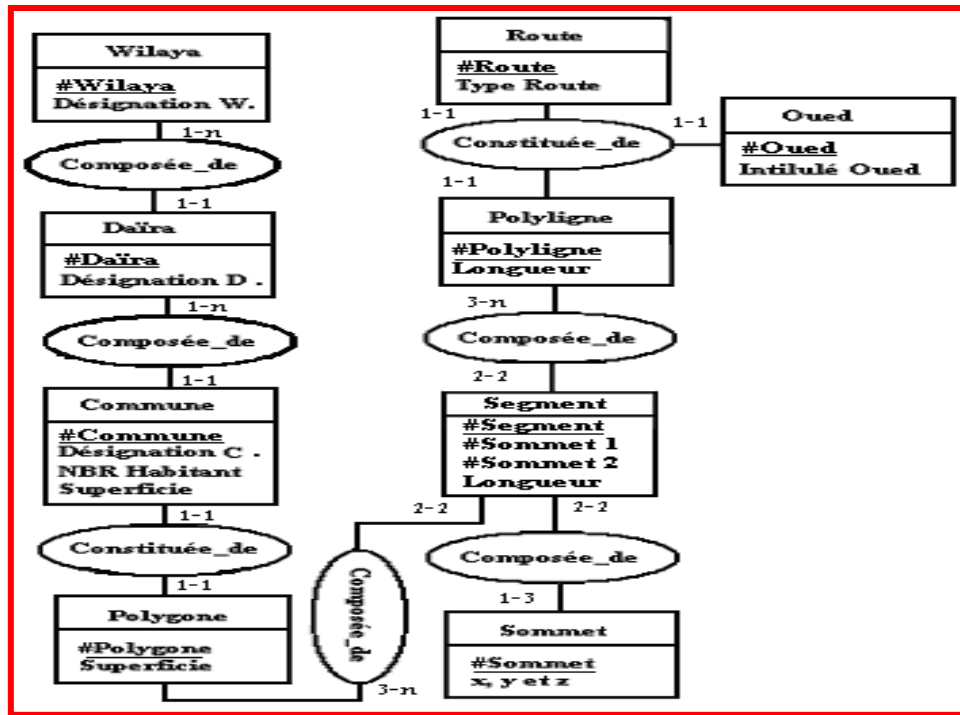


Figure XVIII.1 – Le modèle conceptuel

## XVIII.2 – La présentation de l’environnement de développement

Pour la réalisation de notre application, on a opté pour l’environnement MapInfo 6.5 pour des raisons suivantes :

- MapInfo est un logiciel, qui structure les informations en tables, où chaque table est un ensemble de fichiers qui sont manipulés par le logiciel pour stocker des données.
- MapInfo est capable de permettre de travailler avec des tables créées dans des systèmes de représentations différents.
- MapInfo permet la connexion implicite aux bases de données relationnelles.

## XVIII.3 – La présentation de l’interface utilisateur

L’installation du logiciel de notre application, par la commande SETUP, va créer un répertoire "Projet-SIGADRAR", dans lequel il y aura chargement de l’application "Projet-SIGADRAR". Il y a deux modes d’accès à notre application : mode utilisateur simple, où l’on a accès uniquement pour des consultations des données et des cartes, et mode concepteur ou administrateur, où l’on a la possibilité de procéder à des consultations, des modifications et des mises à jour des cartes et de la base de données via notre interface, des changements de la structure de la base de données, etc. comme sur la figure XVIII.2.

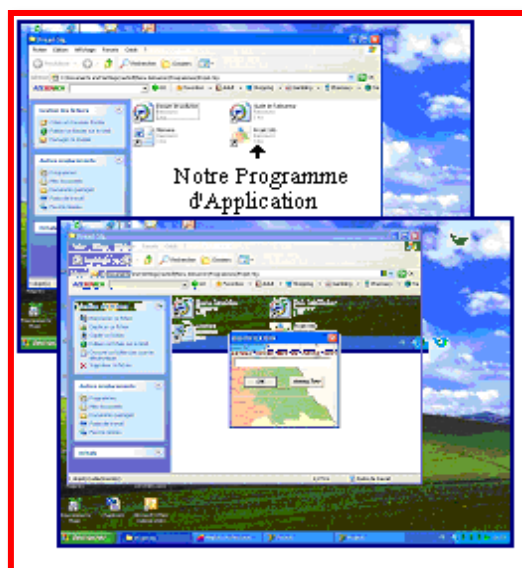


Figure XVIII.2 – L'interface utilisateur

#### XVIII.4 – Les manipulations des données géographiques de population et réalisation

Dans notre exemple, si un utilisateur désire faire une opération d'ajout sur la base de données géographiques (la carte) d'un nouveau projet d'aménagement, alors son insertion au sein des composants de la carte est faite par le SGBD géographique. Il permet l'extraction de toutes les informations dont on a besoin comme sur la figure XVIII.3.

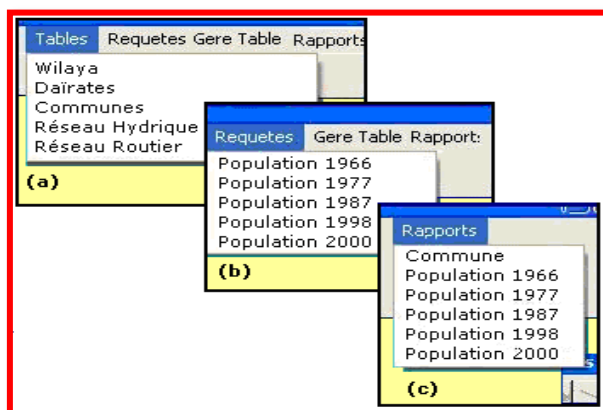


Figure XVIII.3 – Les menus de notre application

##### XVIII.4.1 – Les requêtes descriptives d'identification et de caractérisation

Les requêtes descriptives sont des interrogations de la base de données descriptives. Elles nous fournissent les données de caractérisation (intitulé, superficie, nombre d'habitants,

densité de population) des objets géographiques telles les wilayas, les daïras et les communes comme sur la figure XVIII.4.

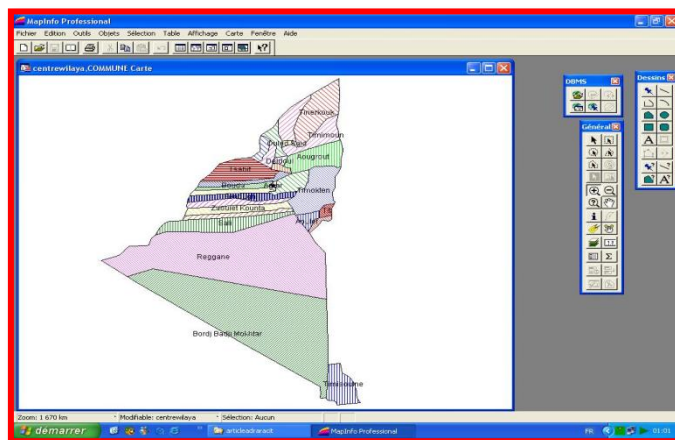


Figure XVIII.4 – Requêtes de caractérisation des objets géographiques

### XVIII.4.2 – Les requêtes descriptives d’identification et de caractérisation

La figure XVIII.5 illustre la répartition des populations durant deux périodes différentes.

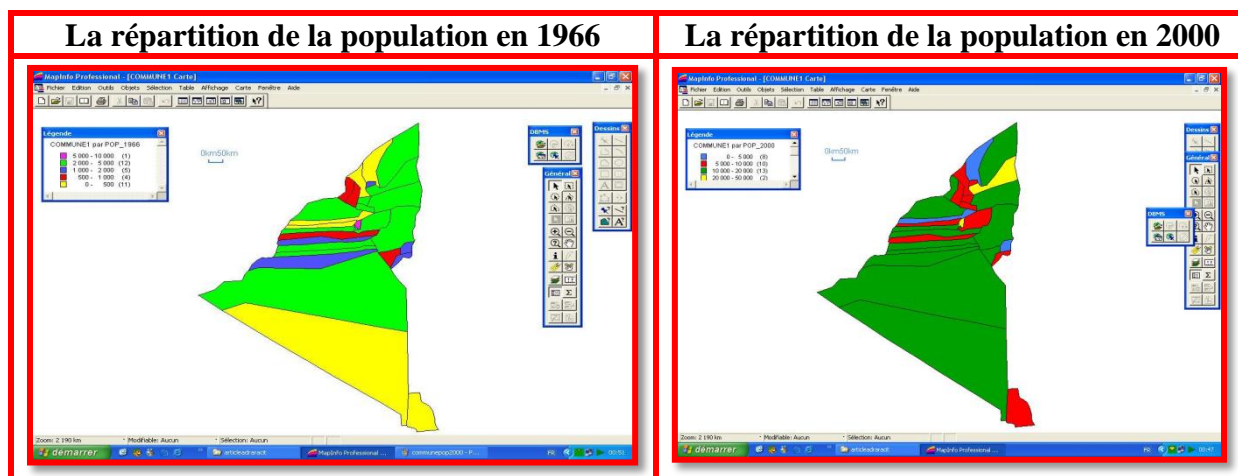


Figure XVIII.5 – La répartition des populations en 1966 et en 2000

### XVIII.4.3 – Les requêtes spatiales

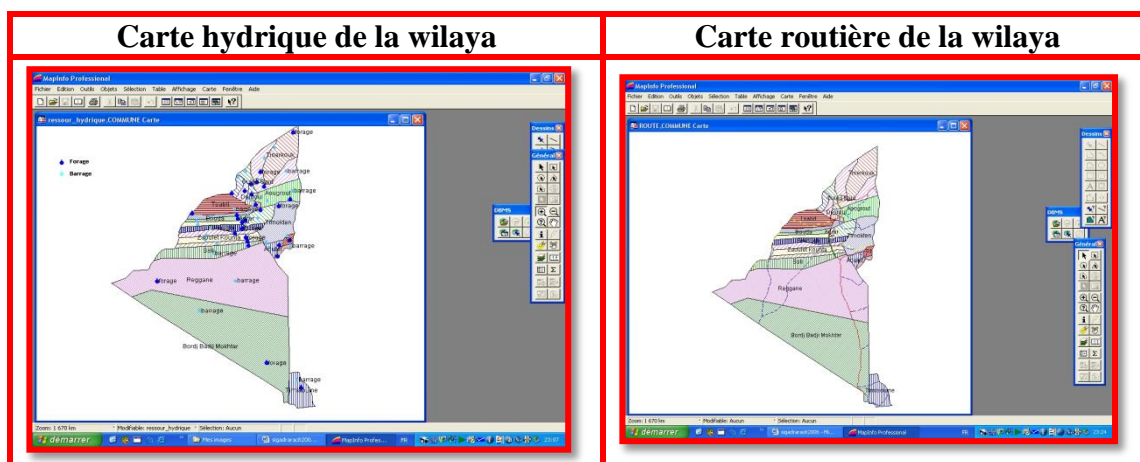
Il existe plusieurs variétés de requêtes. Pour nous, il s’agit des requêtes suivantes :

1. Requêtes de localisation et d’identification : Elles permettent d’extraire toutes les informations concernant un objet qui est représenté sur la carte, ceci revient à poser la question "qu’est ce qu’il y a à cette endroit ?" ou "où suis-je ?".

2. Requête de régionnement (ou de zonage) : C'est le même principe de la requête d'identification, seulement là, on demande des informations concernant une région sélectionnée sur une carte par l'utilisateur.
3. Requête de distance : Cette requête permet de trouver la distance entre des sommets pointés sur la carte. Du point de vue réalisation, nous avons développé plusieurs requêtes SQL et rapports concernant la population : l'évolution de la population de la wilaya d'Adrar de 1966 à 2000, le réseau routier, le réseau hydrique, etc.

### **XVIII.4.3 – Les requêtes spatiales**

La figure XVIII.6 illustre les requêtes spatiales sur des cartes hydrique et routière de la wilaya d'Adrar.



*XVIII.6 – Les cartes hydrique et routière de la wilaya d'Adrar*

### **XVIII.5 – La conclusion et les perspectives**

L'étude de l'évolution de l'implantation des populations de la wilaya d'Adrar est un cas parmi tant d'autres au grand Sud du pays surtout autour des énormes espaces désertiques tels Tamanrasset, Adrar, Tindouf, etc. La mauvaise occupation des sols par des populations arrive à des seuils critiques. Les pouvoirs publics veulent encourager les populations à s'implanter et à créer des projets d'aménagement et de développement urbains permettant de les fixer dans le grand Sud à travers des mesures incitatives.

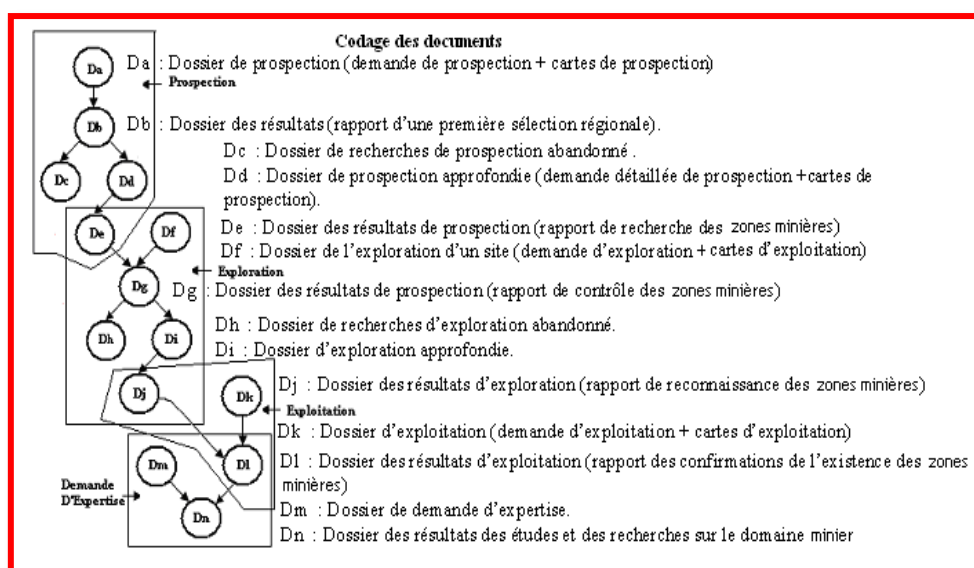
### **XIX – Le cas n°3: Système complexe SIG pour le suivi et la gestion des données de la géologie minière de la région de Constantine**

Pour pouvoir résoudre les problèmes complexes liés à notre cas, on a eu recours à l'outil informatique qui sera capable de :

1. réduire les recherches en améliorant les coûts de fonctionnement par l'automatisation des cartes ;
2. mémoriser les données textuelles qui sont attribuées aux objets graphiques du plan ;
3. structurer les données désagrégées de façon à permettre aux responsables locaux d'extraire les informations mémorisées qui les aident à la prise de décisions et simplifier les modifications qui peuvent être faites sur ces données collectées.
4. les structurer et les stocker dans une base de données spatiale via un interface homme machine dédié.

### **XIX.1 – Le découpage en processus de l'application suivi de la géologie minière**

La figure XIX.1 illustre le découpage en processus de l'application du suivi de la géologie minière de la wilaya de Constantine.



*Figure XIX.1 – Découpage en processus de l'application*

### **XIX.2 – La modélisation du système informatique**

Les acteurs sont les agents identifiés pendant l'étape de la modélisation du système d'information. Pour déterminer les acteurs du système informatique, on doit répondre aux questions suivantes :

1. Quels sont les utilisateurs, qui ont besoins du système pour réaliser leur travail ?
2. Quels sont les utilisateurs, qui vont exécuter les fonctions principales du système ?

3. Quels sont les utilisateurs, qui vont exécuter les fonctions secondaires du système (maintenance et administration) ?
4. Est-ce que le système interagit avec du matériel ou d'autres logiciels ?

Dans notre cas, les acteurs sont :

- Administrateur système : Il gère le processus d'authentification et la définition des profils utilisateurs.
- Spécialiste du domaine : Il consulte la base de données standard, édite les cartes, met à jour la base de données et édite des rapports.
- Décideur : A le droit de consulter toute la base de données y compris les cartes privées et d'éditer des rapports.
- Usager public : A le droit de consulter uniquement les cartes et les rapports standard sans faire aucune modification.

### XIX.3 – Le diagramme de contexte

Les messages sont représentés sur chaque lien. Ils peuvent être décrits d'une façon textuelle pour ne pas surcharger le diagramme de la figure XIX.2.

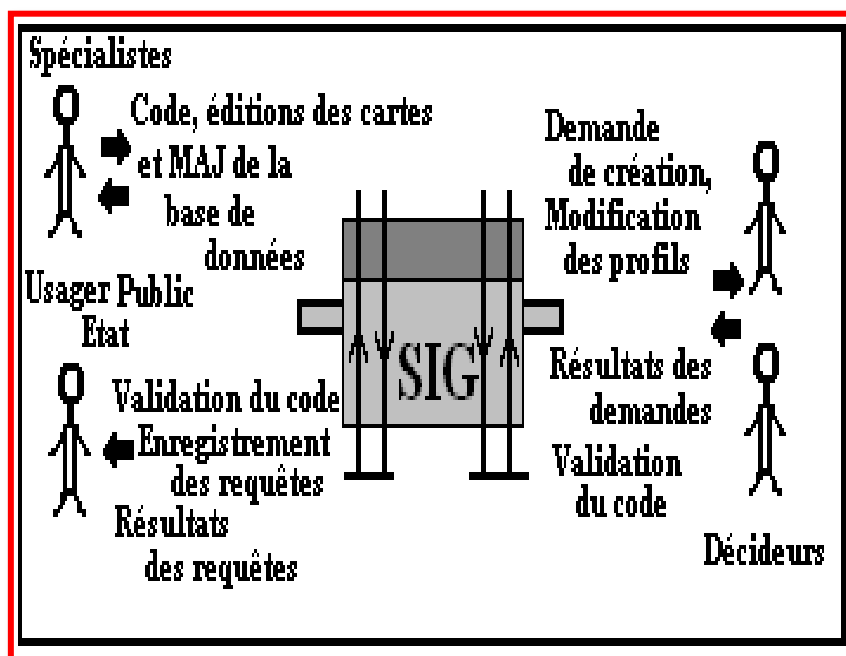


Figure XIX.2 – Diagramme de contexte de notre application

### XIX.4 – La liste préliminaire des cas d'utilisation

Le tableau suivant illustre le diagramme des cas d'utilisation de notre application selon la méthode UML.

<b>Cas d'utilisation</b>	<b>Acteur Principal Acteur Secondaire</b>	<b>Messages envoyés Messages reçus</b>
Authentification	Système	Emet : Validation des codes Reçoit : Nom de l'utilisateur et le code
Editer les cartes Mettre à jour la base de données	Spécialistes du domaine	Emet : Saisie du code Création de la BD Modification de la BD Éditions des cartes. Reçoit : Message système de validation du code Enregistrement de données.
Consulter les cartes standard	Spécialistes du domaine Décideurs Usagers publics	Emet : Envoi des requêtes Consultation des cartes Reçoit : Résultats des requêtes
Consulter les cartes privées	Décideurs	Emet : Saisie de code Envoi des requêtes Reçoit : Message système de validation du code. Consultation des cartes Privées. Résultats des requêtes
Editer les rapports	Spécialistes du domaine Décideurs	Emet : Saisie du code, Envoi de demande d'éditer les rapports. Reçoit : Message système de validation du code Résultat de la demande

### **XIX.5 - Le diagramme des classes**

La figure XIX.3 illustre le diagramme de classes de notre application.



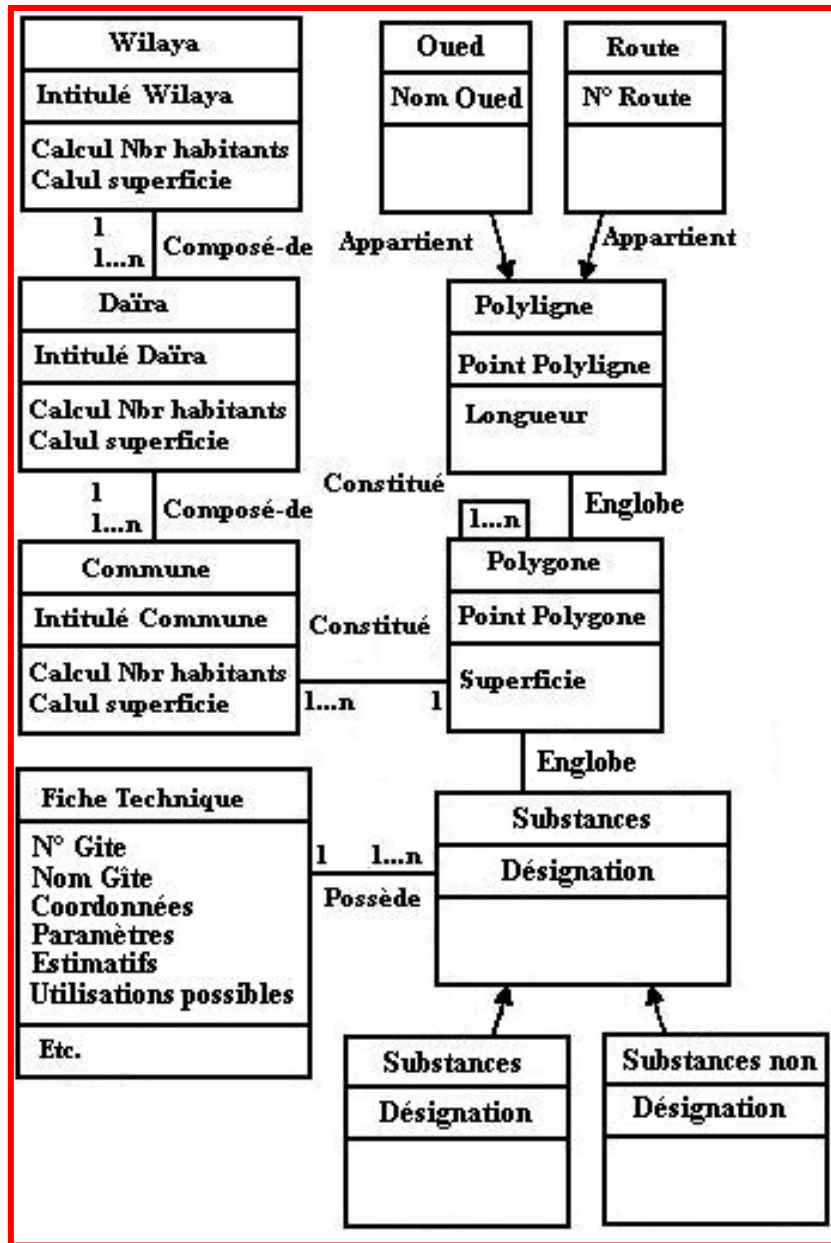


Figure XIX.3 – Le diagramme des classes de notre application

## XIX.6 – La réalisation

On va présenter l'environnement matériel et logiciel dans lequel l'application est développée. Le VB est un outil de programmation orienté objet, il offre une efficacité très remarquable pour le développement de tout genre d'applications, les bibliothèques de classes fournies avec ce langage rendent VB très puissant, sa portabilité sur toutes les plates formes et les systèmes d'exploitations est l'un de ses grandes atouts. Le MapInfo avec environnement de développement la version 6.5.

### XIX.6.1 – La fenêtre "Information"

La figure XIX.4 illustre notre fenêtre d'information de notre application.

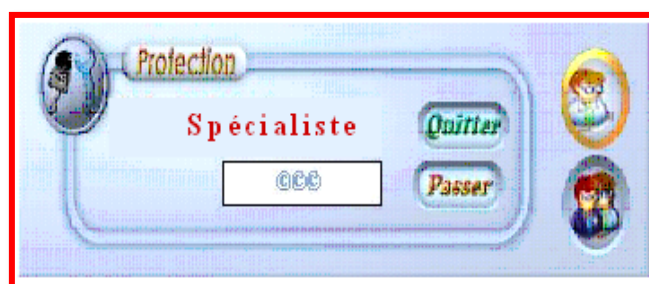


*Figure XIX.4 – Fenêtre d'information*

### XIX.6.2 – La fenêtre "Mot de passe"

On a développé deux sessions comme suit et sur la figure XIX.5 :

- Session Spécialiste : Le spécialiste a le droit d'apporter des modifications à la base de données en ajoutant/supprimant des données, créant des nouvelles couches et en enregistrant ces modifications, c'est pour ça la fenêtre de travail dans la session spécialiste offre toutes les possibilités de consultation et de mise à jour.
- Session Utilisateur : L'utilisateur n'a le droit qu'à la consultation de la base de données, pour empêcher l'utilisateur à effectuer des modifications. Quelques options dans les menus sont désactivés tel que :
  - Nouveau et Enregistrer sous : dans le menu fichier.
  - Tout le menu table : importer, exporter et gestion de table (renommer, modifier structure, supprimer et compacter).



*Figure XIX.5 – La fenêtre du mot de passe*

### XIX.6.3 – la fenêtre de travail (principale)

Elle contient six parties comme sur la figure XIX.6 :

1. La barre de "menus" ;
2. Barre d'outils "9 boutons" ;
3. Barre d'outils "requête et rapport" ;
4. Partie d'affichage "Nom de cartes ouvertes" ;
5. Partie d'affichage "Carte géographique" ;
6. Et partie d'affichage "Propriétés et tables de données".

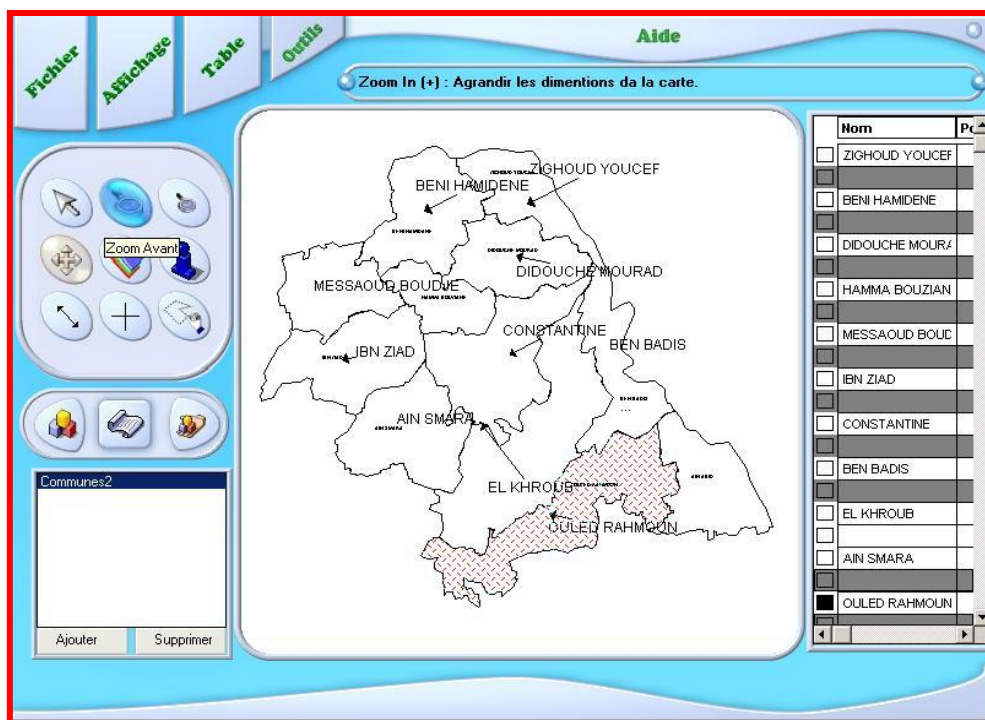


Figure XIX.6 – La fenêtre principale de travail sur notre application

### XIX.6.4 – La fenêtre "Boîte de message d'erreur"

Cette fenêtre s'affichera au centre de l'écran, et elle se fermera en effectuant un click le bouton "fermer" comme sur la figure XIX.7.



Figure XIX.7 – La fenêtre "boîte de message d'erreur"

### **XIX.6.5 – Les requêtes**

On a deux types de requêtes de description et spatiales.

Les requêtes descriptives d'identification et de caractérisation

- Les requêtes descriptives sont des interrogations de la base de données descriptives

Les requêtes spatiales

Il existe plusieurs variétés de requêtes. Pour nous, il s'agit des requêtes suivantes :

1. Requêtes de localisation et d'identification : Elles permettent d'extraire toutes les informations concernant un objet qui est représenté sur la carte, ceci revient à poser la question "qu'est ce qu'il y a à cette endroit ?" ou "où suis-je ?".
2. Requête de régionnement (ou de zonage) : C'est le même principe de la requête d'identification, seulement là, on demande des informations concernant une région sélectionnée sur une carte par l'utilisateur.
3. Requête de distance : Elle permet de trouver la distance entre des sommets pointés sur la carte.

### **XIX.7 – La conclusion et les perspectives**

On a bénéficié de la richesse d'UML, exprimée par ses diagrammes et ses concepts qui offrent une clarté et une précision indispensables pour tout concepteur et qui représentent un moyen performant de modélisation, de documentation et de communication. Le processus 2TUP, qui nous a servi comme un fil d'Ariane tout au long de notre travail, nous a permis de bien maîtriser le développement de notre application, et également d'exploiter le maximum les concepts qu'offre UML. Le bilan de notre travail est dans une large mesure atteint, puisque l'environnement est disponible pour permettre des consultations des données géographiques de la géologie minière de la région de Constantine.

Il reste bien sûr à développer des modeleurs permettant des simulations par les spécialistes et les chercheurs universitaires pour qu'ils puissent simuler leurs scénarios de prospection et d'exploitation avec des outils de nouvelles technologies d'information et de communication et des algorithmes d'explorations des solutions plus efficaces et optimales.

## **XX – La conclusion générale de la présentation des applications en Algérie**

Présentation de trois cas d'études d'application des systèmes complexes SIGs dans des domaines très divers. Il reste à aborder des cas d'étude, que nous avons développés avec nos étudiants des filières d'ingénieur, de Magister et de Doctorat, sur :

1. Cas de gestion et du suivi de l'avancé du désert dans le région de Biskra (problème de sécheresse);
2. Cas de la gestion et du suivi de la pollution de la région de Constantine;
3. Cas de glissements de terrain dans la région de Constantine;
4. Cas de la gestion des fougates dans la région d'Adrar;
5. Cas de la gestion minière de la région de Constantine;
6. Etc.