MOIA - Troisième Partie : Planification - Apprentissage - Méthodes incomplètes - Linguistique

F. Bouquet

Master S&T - Mention Informatique

Première année

Problèmes d'ordonnancement

But : *fixer les dates de démarrage des tâches* de façon à *respecter les contraintes* selon un *critère objectif à optimiser*.

Données du problème :

- Ensemble de *tâches* $I = \{t_1, t_2, \ldots t_n\}$
- Chaque tâche t_i possède une *durée* d_i
- Ensemble de contraintes C sur I

Les contraintes & Critères

Type de contraintes :

- Contrainte de **précédence** :
  - $prec(i, j)$: la tâche i doit être terminée avant de pouvoir commencer l'exécution de j.
- Contrainte de **ressources** :
  - ressources consommables
  - ressources renouvelables : disjonctives, cumulatives

Critères d'optimisation :

- minimisation de la durée totale
- minimisation du nombre de tâches en retard
- minimisation des consommables . . .

Exemple : Job-Shop

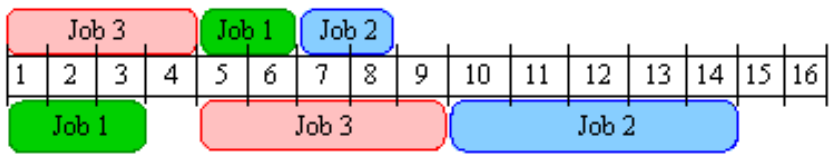
- **Données :**
  - m machines
  - n séquences d'opérations ($\equiv$ jobs)
- **Contraintes :**
  - Ressources = machines (contrainte disjonctive)
  - Précédence = jobs
- **Solution :** Ordre d'exécution des opérations des jobs sur chaque machine
- **Hypothèse :** Un job ne passe qu'une fois par machine

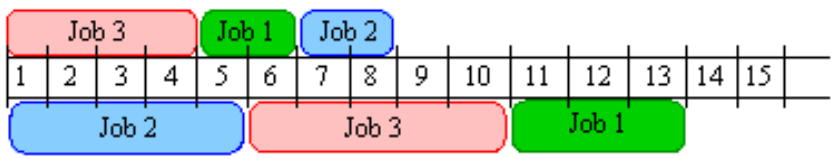
$$\Rightarrow \text{nombre de solutions : } (n!)^m$$

Job-Shop 3 × 2

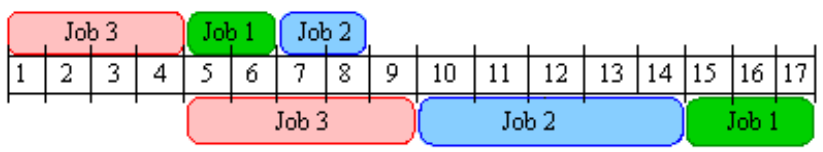
Machine	Job 1	Job 2	Job 3
M_1	2	2	4
M_2	3	5	5

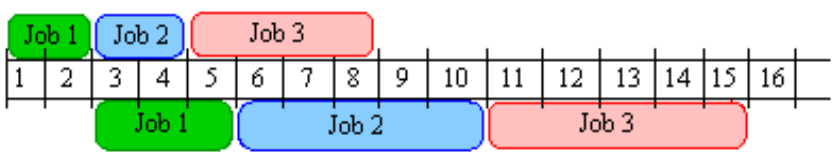
- Sans ordre sur les machines
  Première : tâche de durée **max**





- Avec ordre sur les machines
  Première : tâche de durée **min**





Contraintes de précédence

Représentation Graphe des potentiels

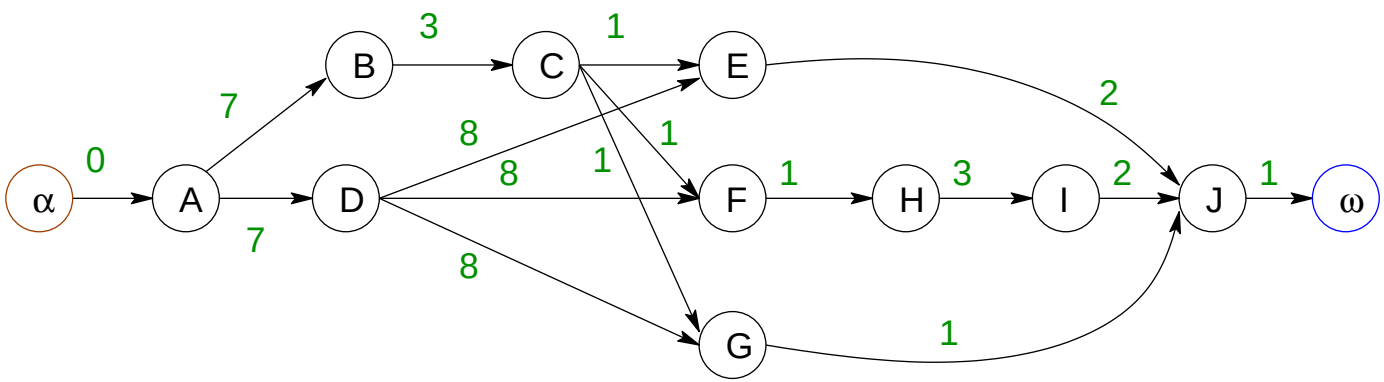
- Sommets : tâches
- Arcs : précédences
- Valuations : durées

Propriétés

- G est sans circuit, décomposable en niveaux
- Résolution d'un chemin de durée maximale
- Date de démarrage au plus tôt
- Date de démarrage au plus tard

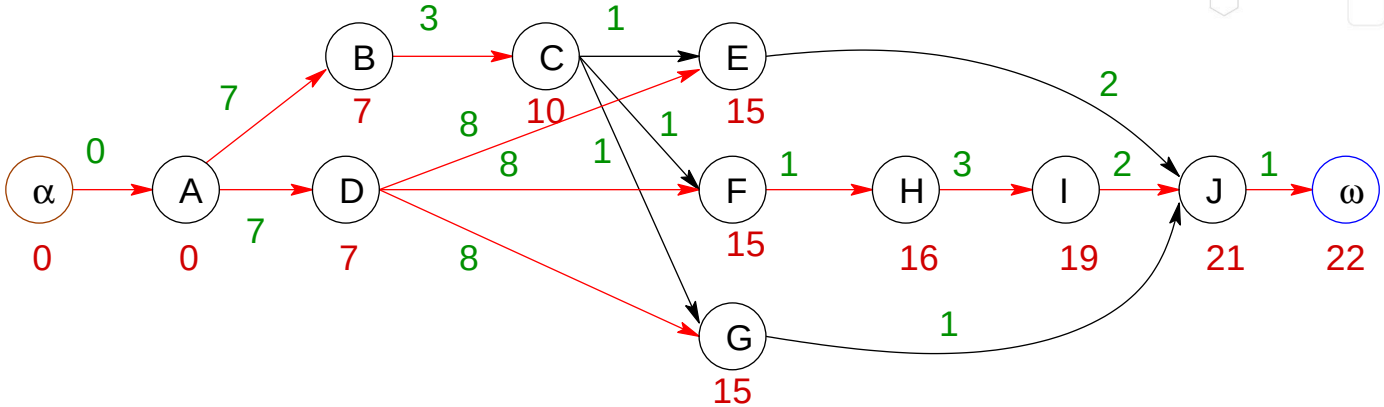
Exemple de graphe

Tâche(t_i)	A	B	C	D	E	F	G	H	I	J
Durée(p_i)	7	3	1	8	2	1	1	3	2	1
Précedence		A	B	A	C, D	C, D	C, D	F	H	E, I, G



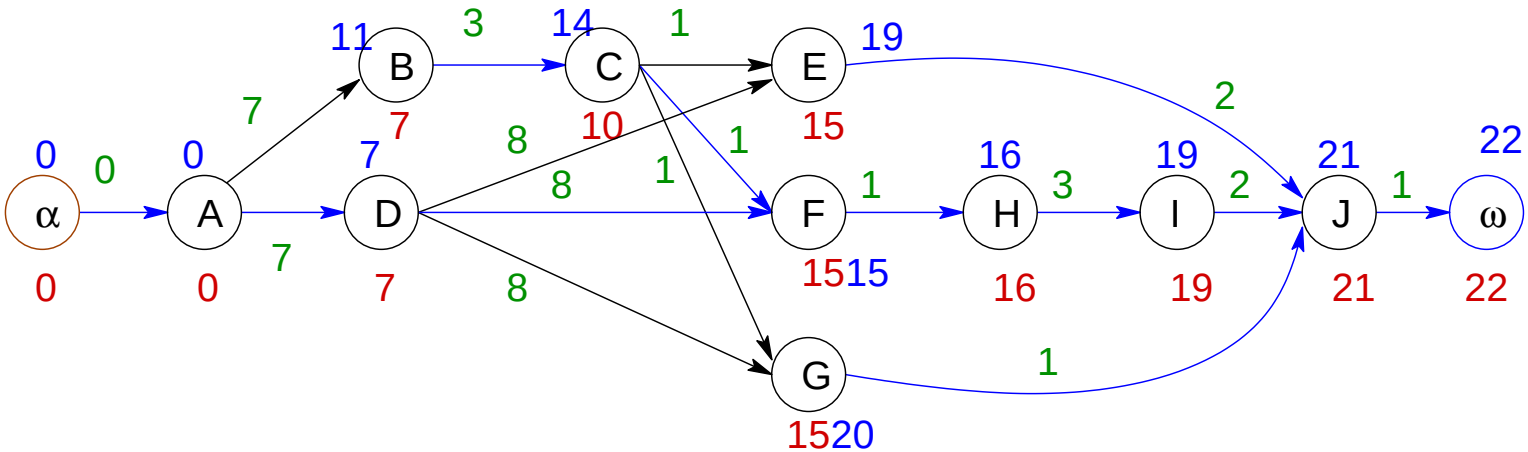
ajout de deux sommets : α et ω

Exemple - Suite



Date de démarrage au plus **tôt** t_i :

$$\begin{cases} t_\alpha & = & 0 \\ t_i & = & max_{j \in \Gamma(i)} \{t_j + p_j\} \end{cases}$$



Date de démarrage au plus **tard** :

$$\begin{cases} T_\omega & = & t_\omega \\ T_i & = & min_{j \in \Gamma^{-1}(i)} \{T_j - p_i\} \end{cases}$$

Apprentissage

IA support pour des fonctions cognitives :

- penser (raisonner)
- communiquer (entendre, voir, parler)
- planifier et agir efficacement
- **apprendre**

Pourquoi ?

- Capacité d'adaptation :
  - base de la survie de l'espèce
  - augmente le contrôle sur l'environnement
- diminue : la précision requise pour opérer
- augmente : adaptabilité, autonomie, flexibilité, la quantité de choses que l'on peut faire.

Apprendre

Apprendre est un signe d'intelligence :

- capacité d'*association* (mémoire)
- capacité d'*abstraction* (génération de concepts)
- *explicabilité* (comportement rationnel)
- capacité de *prévision* (anticipation : + contrôle)
- *réplicabilité* (- aléatoire dans le comportement)
- capacité de *révision* (dynamicité), capacité d'*adaptation*...

Pourquoi apprendre ? Apprendre sur l'apprentissage

- Pas algorithmes précis : reconnaissance visages, voix ...
- Trop de données à traiter : "data mining", "market analysis"...
- Données "volatiles" : prédiction de réactions (météo)

Exemples d'applications

- **analyse de données** : des marchés financiers, banques
- **analyse de comportements** : de traitements médicaux, bio-informatique
- **contrôle et optimisation de réseaux de distribution** : d'électricité, d'eau, de services de (télé-)communications, de trafic urbain (aérien), de dépenses énergétiques d'un immeuble ou d'une habitation
- **personnalisation de services** : personnalisation des interfaces (TV interactive)
- **identification d'individus** : écriture (signature), voix, empreintes digitales et rétiniennes, visage...

Définition

"Un programme d'ordinateur est capable d'apprendre à partir d'une expérience E et par rapport à un ensemble T de tâches et selon une mesure de performances P, si sa performance à effectuer une tâche T mesurer par P s'améliore avec l'expérience E."

(T. Mitchell, Machine Learning McGraw Hill, 1997)

exemple : apprendre à jouer aux échecs

- T : jouer aux échecs et gagner
- P : % de victoires
- E : possibilités de jouer contre quelqu'un

Apprentissage et inférences

- apprentissage **"par cœur"** : répéter comme un perroquet ce que l'on vous a dit.
- inférence **déductive** : à partir de A et de $A \rightarrow B$ inférer B.
- inférence **abductive** : à partir de B et de $A \rightarrow B$ inférer A.
- inférence **inductive** :

- "j'ai été malade après avoir mangé trop de gâteaux de Noël"
- "j'ai été malade après avoir mangé trop de gâteaux de Pâques"

$\Rightarrow$ "je serai malade si je mange trop de gâteaux"

Attention aux généralisations hâtives ! ! !

Types d'apprentissage

- **Supervisé :** l'apprenant reçoit des exemples d'apprentissage comprenant à la fois les valeurs d'entrée et de sortie.
- **Non-supervisé :**
  - l'apprenant reçoit des exemples d'apprentissage ne comprenant que les valeurs d'entrée (pas de valeurs de sortie).
  - l'apprenant peut recevoir une indication en retour qui l'aide à déterminer s'il est dans la bonne direction, mais cette indication est souvent vague, ambiguë ou tardive,
- **"Par paquets" ("batch") :** processus dans lequel des exemples d'apprentissage sont pris en compte tous à la fois.
- **"En ligne" :** processus dans lequel des exemples d'apprentissage sont pris en compte un à un.

Définition

- Soit X l'ensemble de tous les exemples possibles
- Une **classe** C est un sous-ensemble de X.
- L'ensemble d'exemples d'apprentissage est $\{(x , y)\}$ où $x \in X$ et $y \in \{0 , 1\}$.
- Trouver une fonction f telle que $f(x) = 1$ si $x \in C$ et $f(x) = 0$ si $x \notin C$,
- On appelle **espace d'hypothèses** H, l'ensemble de toutes les fonctions f possibles.
- On peut généraliser de $H = \{ f : X \to \{0 , 1\}\}$ à $H = \{ f : X \to Y \}$ où X et Y sont des espaces de valeurs d'entrée et de sortie arbitraires.

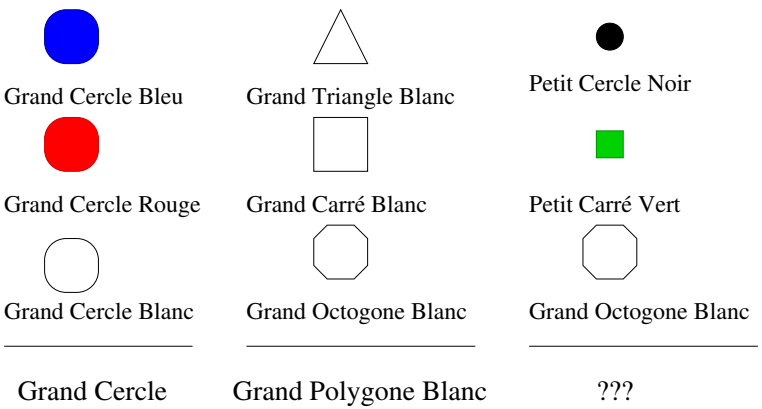
Biais

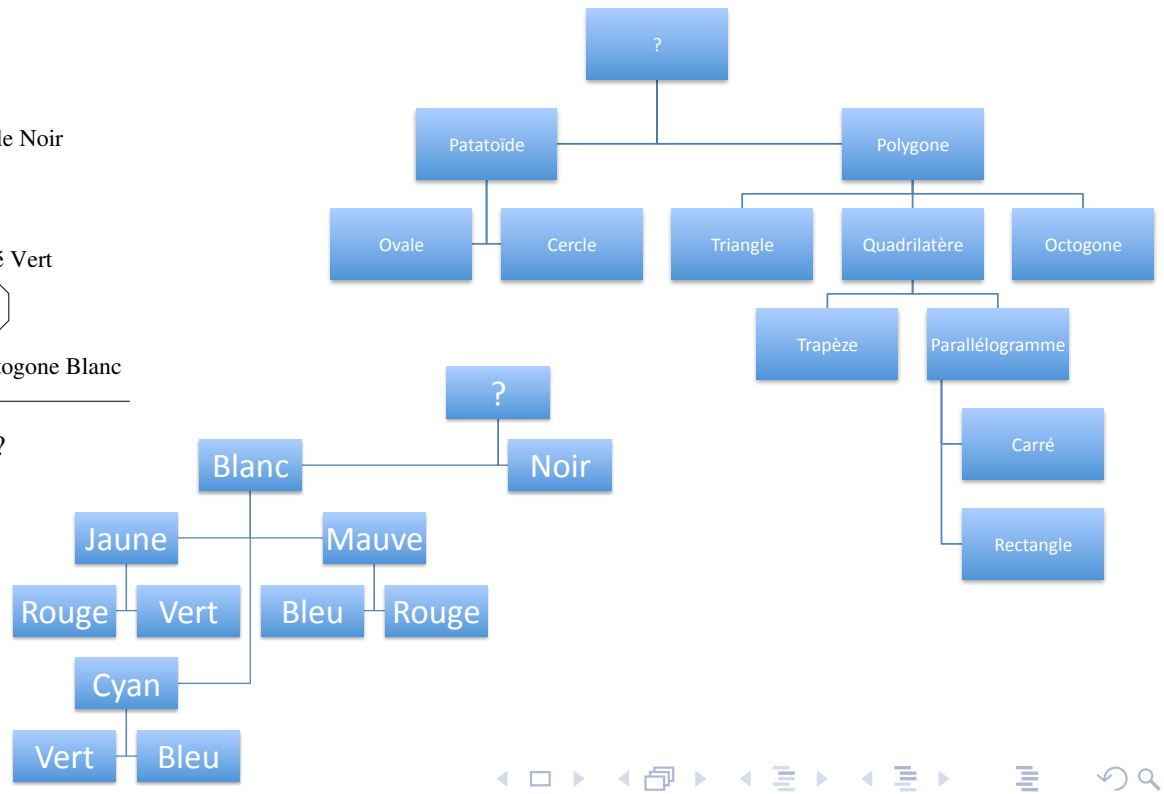
- **Biais inductifs** fournit à l'apprenant une base pour choisir parmi les fonctions de H.
- **Biais restrictif** choisir un ensemble restreint de solutions possibles :
  - l'ensemble des fonctions linéaires de $\mathbb{R}$ dans $\mathbb{R}$,
  - l'ensemble des conjonctions de N variables logiques.
- **Biais de préférence** détermine un ordre sur l'ensemble des solutions possibles :
  - préférer les formules logiques courtes,
  - préférer les polynômes de faibles degrés.

Généralisation

Une hypothèse possède la propriété de généralisation si elle s'applique aussi bien à des données nouvelles qu'aux exemples d'apprentissage.

- Généraliser un concept c'est, pour un concept C, trouver un concept plus large $C \subset C'$.
- Spécialiser un concept c'est, pour un concept C, trouver un concept plus étroit $C' \subset C$.





Méthodes d'apprentissage

- Systèmes à base de règles
- Réseaux de neurones artificiels
- Arbres de décision
- "Mélanges" d'experts
- Théorie bayésienne de la décision
- Techniques de groupement ("clustering")
- ...

Sur-apprentissage

"Mémorisation de tout l'ensemble d'apprentissage plutôt que d'induire un concept général".

- Plus on apprend, plus on "colle" aux données
- A la fin tous les exemples sont appris par cœur, y compris le bruit : phénomène de sur-apprentissage
- Le critère d'arrêt de l'apprentissage doit éviter ce piège
- Soit $erreur_{app}(h)$ l'erreur commise par l'hypothèse h sur les données d'apprentissage, et $erreur_D(h)$ l'erreur commise par h sur la distribution totale des données :

  Définition : On dira que h sur-apprend les données d'apprentissage si il existe une hypothèse h' telle que : $erreur_{app}(h) < error_{app}(h')$ et $error_D(h) > error_D(h')$

Corriger le sur-apprentissage

- Arrêt aussitôt que possible :
  - cesser d'étendre lorsque le partitionnement n'est plus significative,
  - test si la partition produite améliore la précision.
- Élagage à posteriori :
  - construire l'arbre entier
  - supprimer les noeuds inutiles (test approprié).

Apprentissage arbre de décision

Construction :

- **Nœud non-terminal** représente un test sur un attribut.
- **Branche** correspond à une valeur possible pour un attribut.
- **Nœud terminal** correspond à une classification possible.

Intérêt :

- Représentation
- Exemples décrits par des paires $<$ attributs, valeurs $>$
- Exemples d'apprentissage sont incomplets ou bruités.
- Fonction-objectif est à valeurs discrètes.

ID3

Algorithme pour la classification par arbre de décision

Entrées : Partie du corpus d'études (entre 40 et 70 %)

Sortie : Un arbre de décision

Validation : Deuxième partie du corpus qui valide les règles

Description : Appel : ID3($Ce, F_{ob}, Attributs$)

- Corpus d'études Ce : ensemble des individus
- Un individu à des *Attributs* : Valeur(Ind,Att)
- Une *fonction-objectif* $\approx$ Attribut : $F_{ob}(Ind) = Obj$
- $set(Attribut, Val)$: ensemble des individus $Attribut = val$

Algorithme ID3

Algorithme 1 : Algorithme : ID3(Ce, F_{ob}, Attributs)

Input : une partie du corpus d'études Ce
Une fonction-objectif F_{ob}
La liste des attributs à traiter Attributs
Output : un arbre de décision
Créer un nœud *n* **if** F_{ob} *est égale à une seule valeur* **then** Etiquette(n) = valeur;
else if *Attributs* $= \emptyset$ **then**
    Etiquette(n) = valeur$_{\mathbf{max}}$ de F_{ob}
else
    Soit A $\in$ Attributs et A l'Attribut le plus discriminant;
    Etiquette(n) $\leftarrow$ A;
    i $\leftarrow$ 0;
    for *chaque Valeur possible de A* **do**
        if *set(A,Valeur)* $\neq \emptyset$ **then**
            Etiquette_arc(n,i) = Valeur ;
            Fils(n,i) = ID3(set(A,Valeur),F_{ob},Attributs $\setminus$ A);
            i $\leftarrow$ i+1;
return n;

Attribut le plus discriminant

Il existe plusieurs critères pour :

- Notion d'entropie : nombre de bits nécessaires pour coder la classification de membres de S choisis aléatoirement avec un code optimal. Entropie(S) $= \Sigma_{i=1}^{Card(F_{ob})} p(i) \times -log_2(p(i))$
- *Table de contingence*, ligne liée à F_{ob} et colonne à S :
  - $n_i = \Sigma_{i=1}^{Card(F_{ob})} set(F_{ob}, i), n_j = \Sigma_{j=1}^{Card(S)} set(S, j)$
  - Critère global de S $= \Sigma_{i=1}^{Card(A)} \text{MAX}_{j \in 1..Card(F_{ob})} Table(i, j)$
  - Information mutuelle de S $= \frac{2 + n_{ij}^{log} - n_i^{log} - n_j^{log} + nb * log(nb)}{nb}$
  - tau de S $= \frac{nb * n_{ij}^{i} - n_j}{nb^2 - n_j}$

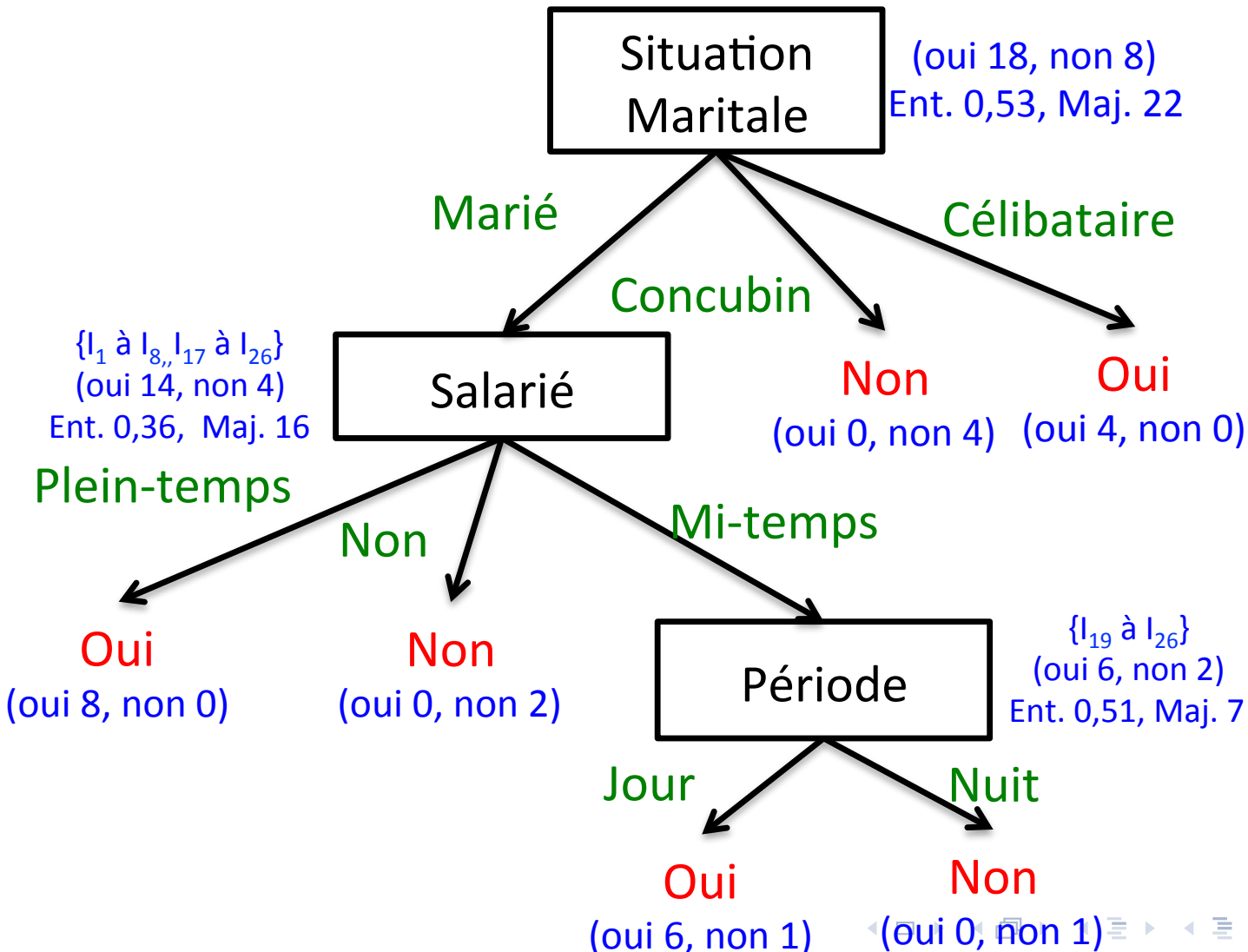
Le choix de l'attribut A le plus discriminant :

- l'entropie : $\text{Min}_A(\Sigma_{i=1}^{Card(A)} \frac{set(\text{Corpus}, A, A(i))}{Card(\text{Corpus})} \times \text{Entropie}(A(i)))$
- le critère majoritaire : $\text{Max}_A(\text{CG}(A))$

Exemple - les données

Individu	Salarié	Situation Familiale	Période de travail	Crédit
I_1	Plein-temps	Marié	Jour	Oui
I_2	Plein-temps	Marié	Jour	Oui
I_3	Plein-temps	Marié	Jour	Oui
I_4	Plein-temps	Marié	Jour	Oui
I_5	Plein-temps	Marié	Nuit	Oui
I_6	Plein-temps	Marié	Nuit	Oui
I_7	Plein-temps	Marié	Nuit	Oui
I_8	Plein-temps	Marié	Nuit	Oui
I_9	Plein-temps	Concubin	Nuit	Non
I_{10}	Plein-temps	Concubin	Jour	Non
I_{11}	Plein-temps	Concubin	Jour	Non
I_{12}	Plein-temps	Concubin	Jour	Non
I_{13}	Plein-temps	Célibataire	Jour	Oui
I_{14}	Plein-temps	Célibataire	Nuit	Oui
I_{15}	Plein-temps	Célibataire	Nuit	Oui
I_{16}	Plein-temps	Célibataire	Jour	Oui
I_{17}	Non	Marié	Jour	Non
I_{18}	Non	Marié	Nuit	Non
I_{19}	Mi-temps	Marié	Nuit	Non
I_{20}	Mi-temps	Marié	Jour	Non
I_{21}	Mi-temps	Marié	Jour	Oui
I_{22}	Mi-temps	Marié	Jour	Oui
I_{23}	Mi-temps	Marié	Jour	Oui
I_{24}	Mi-temps	Marié	Jour	Oui
I_{25}	Mi-temps	Marié	Jour	Oui
I_{26}	Mi-temps	Marié	Jour	Oui

Exemple


Situation Maritale
(oui 18, non 8)
Ent. 0,53, Maj. 22
Marié
Concubin
Célibataire
{I1 à I8,, I17 à I26}
(oui 14, non 4)
Ent. 0,36, Maj. 16
Salarié
Non
(oui 0, non 4)
Oui
(oui 4, non 0)
Plein-temps
Non
Mi-temps
Oui
(oui 8, non 0)
Non
(oui 0, non 2)
Période
{I19 à I26}
(oui 6, non 2)
Ent. 0,51, Maj. 7
Jour
Nuit
Oui
(oui 6, non 1)
Non
(oui 0, non 1)

Réseau de neurones

- nombre de neurones dans le cerveau $\approx 10^{11}$
- nombre de connexions par neurone $\approx 10^4 - 10^5$
- temps de cycle (switching time) $\approx 10^{-3}$ seconde
- temps moyen d'une activité cognitive ≈ 0.1 seconde

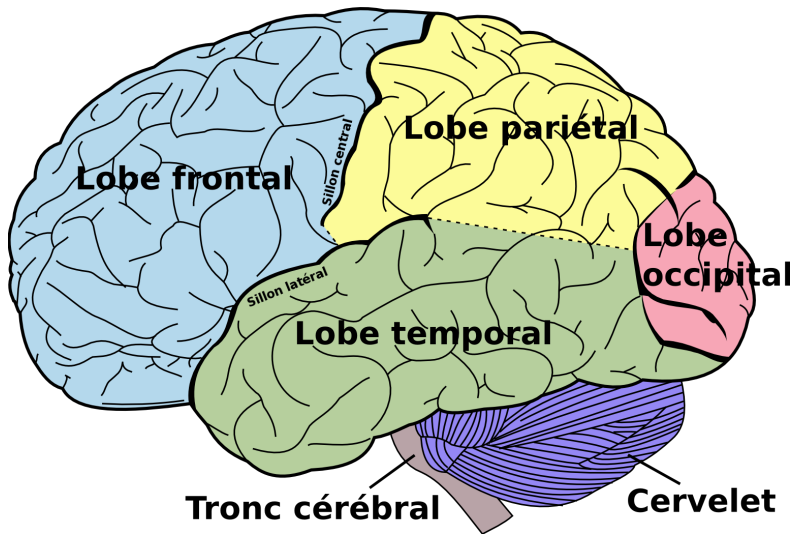
L'**intelligence** : *c'est un millier de connexions pour chacun des 100 milliards de neurones du cerveau humain.*

Mais : traitement distribué et parallélisme massif

- beaucoup d'interconnexions entre unités,
- ajustement automatique des poids des connexions,
- traitement sous-symbolique : intelligence comme propriété émergente

Réseau de neurones - Le cerveau humain

- Composés de plusieurs éléments
- La conscience doit être peu sollicitée...
- Filtre(s) et biais cognitif(s) :
  - Simplification
  - Champs visuel
  - Importance des sens
  - Effet d'amorçage



Réseau de neurones - Historique

- 1906 : Santiago Ramón y Cajál & Camilo Golgi : prix nobel **neurones** éléments de structure du cerveau.
- 1943 : J. McCulloch & W. Pitts : proposent un modèle neuronal simple capable de produire une machine de Turing
- 1948 : D. Hebb : propose une règle d'apprentissage pour des réseaux de neurones
- 1958 : F. Rosenblatt : propose le modèle du *Perceptron* et démontre son théorème de convergence
- 1969 : M. Minsky & S. Papert : démontrent les limitations du modèle du *Perceptron*,
- 1985 : apprentissage par rétro-propagation pour les réseaux multi-couches.

Apprentissage

Modification poids : $\Delta W_{ij} = \eta(T_j - O_j)I_i = \eta\delta_j I_i$

- η : représente le coefficient d'apprentissage
- T : le vecteur théorique en sortie
- O : le vecteur calculé par le réseau en sortie
- I : le vecteur en entrée.
- Variation du poids / signal d'erreur : $\Delta W_{ij} = \eta\delta_j O_i$
- Couche de sortie : $\delta_j = (T_j - O_j)f'_j(N_j)$
- Rétro-propagation couches cachées : $\delta_j = f'_j(N_j)\Sigma_k \delta_k W_{kj}$

Perceptron

Théorème de représentation *Un réseau "feedforward" à une seule couche (Perceptron) peut uniquement représenter des fonctions linéairement séparables. C'est-à-dire celles pour lesquelles la surface de décision séparant les cas positifs des cas négatifs est un (hyper-)plan.*

Théorème d'apprentissage *Étant donné suffisamment d'exemples d'apprentissage, il existe un algorithme qui apprendra n'importe quelle fonction linéairement séparable.*

Convergence assurée car l'erreur E est une forme quadratique dans l'espace des poids. Elle possède donc un seul minimum global et la descente du gradient garantit de la trouver, **Sinon :**

- les données d'apprentissage sont linéairement séparables
- le taux d'apprentissage η est suffisamment petit
- il n'y a pas d'unités "cachées"

Réseau multi-couches

- représenter des fonctions non linéairement séparables :
  Un réseau à 2 couches (une couche cachée) avec des unités à seuil peut représenter la fonction logique xor.
- Les réseaux multi-couches "feedforward" peuvent être entraînés par rétro-propagation pour autant que la fonction de transition des unités soit différentiable (les unités à seuil ne conviennent donc pas).

Différents types de fonctions :

Seuil

$$\varphi(v) = \begin{cases} 1, \text{ si } v \geq 0 \\ 0, \text{ si } v < 0 \end{cases}$$

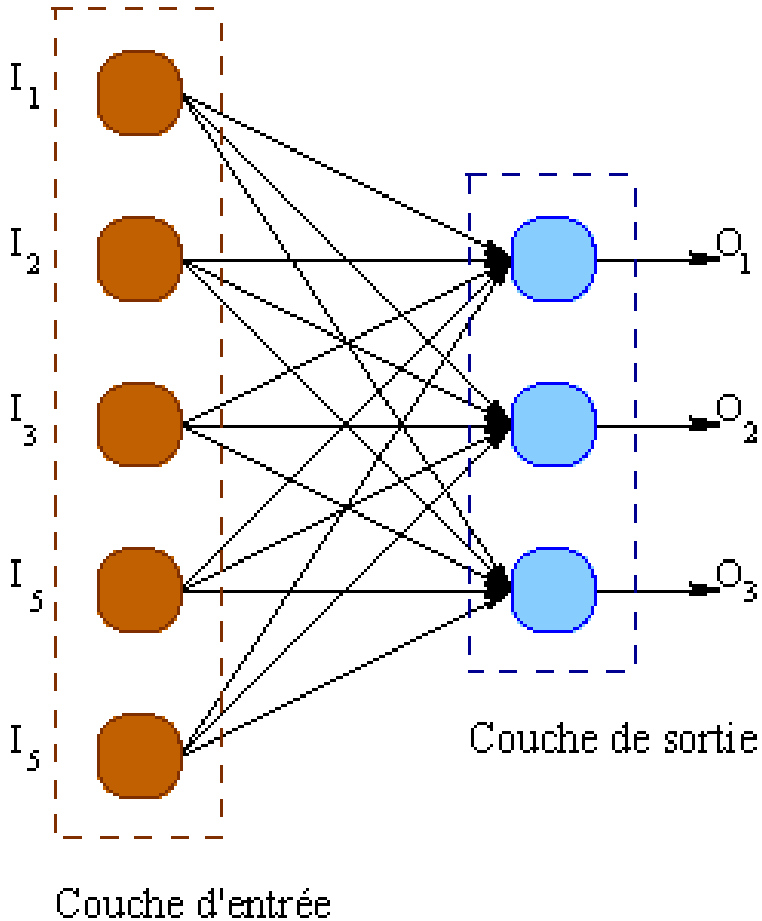
linéaire par partie

$$\varphi(v) = \begin{cases} 1, \text{ si } v \geq \alpha \\ v, \text{ si } \alpha > v \geq \beta \\ 0, \text{ si } v > \beta \end{cases}$$

sigmoïde

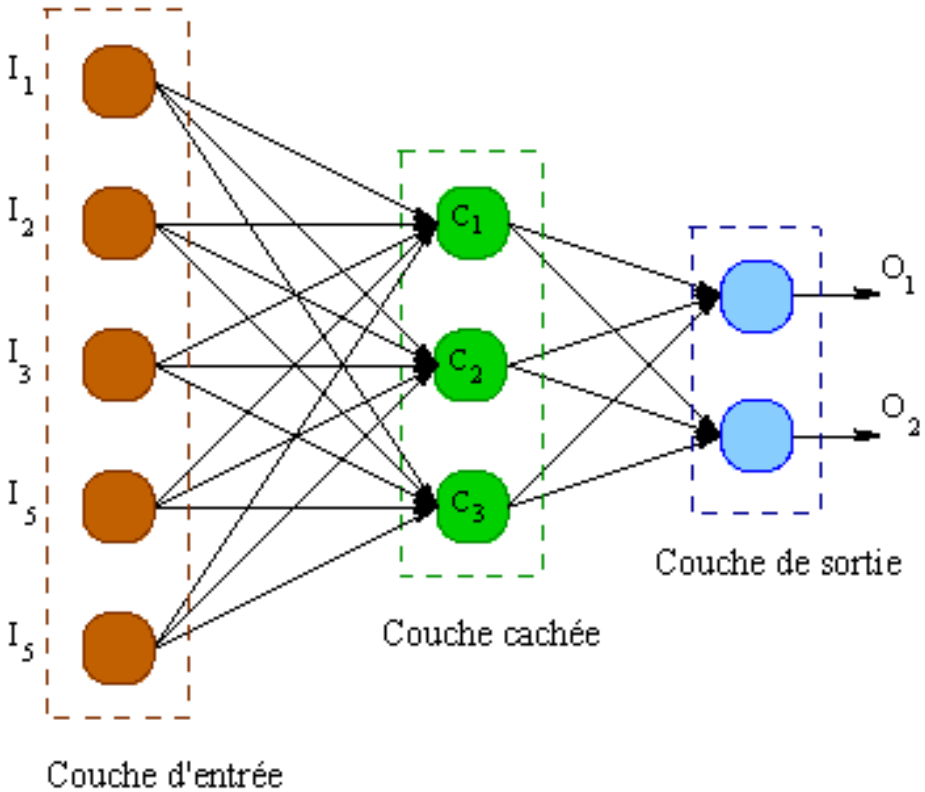
$$\varphi(v) = \frac{1}{1+e^{-\alpha v}}$$
$$\varphi(v) = \frac{1-e^{-v}}{1+e^{-v}}$$

Réseau à une couche



$$O_i = 1 \quad \text{si } \Sigma_k w_{ik} I_k > 0$$
$$O_i = 0 \quad \text{sinon}$$

"Feedforward" ou multi-couches



Couche cachée

$C_j = 1$ si $\Sigma_k w_{jk} I_k > 0$

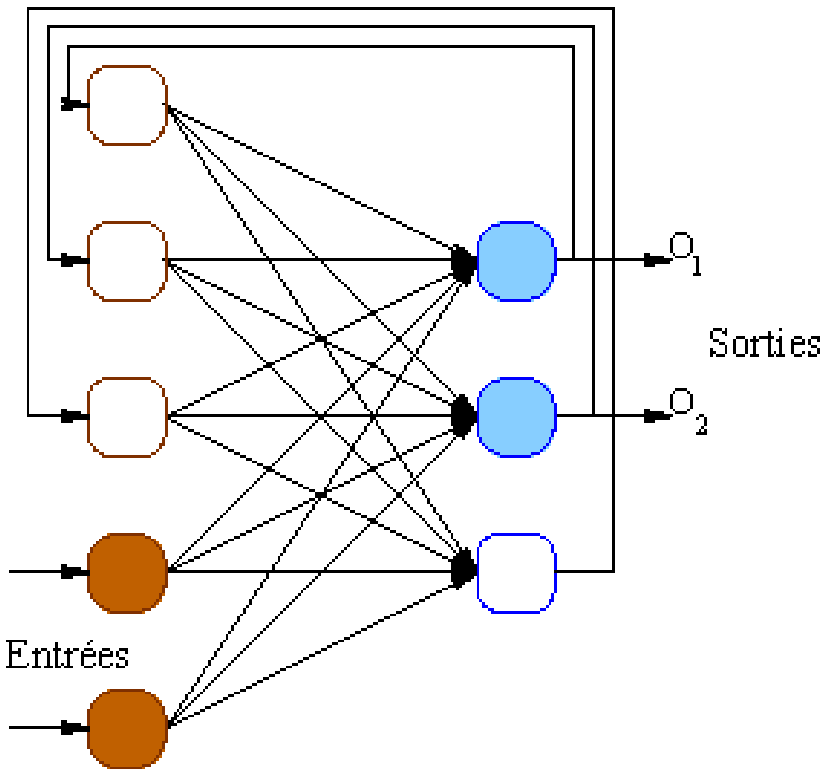
$C_j = 0$ sinon

Couche de sortie

$O_i = 1$ si $\Sigma_k w_{ik} C_k > 0$

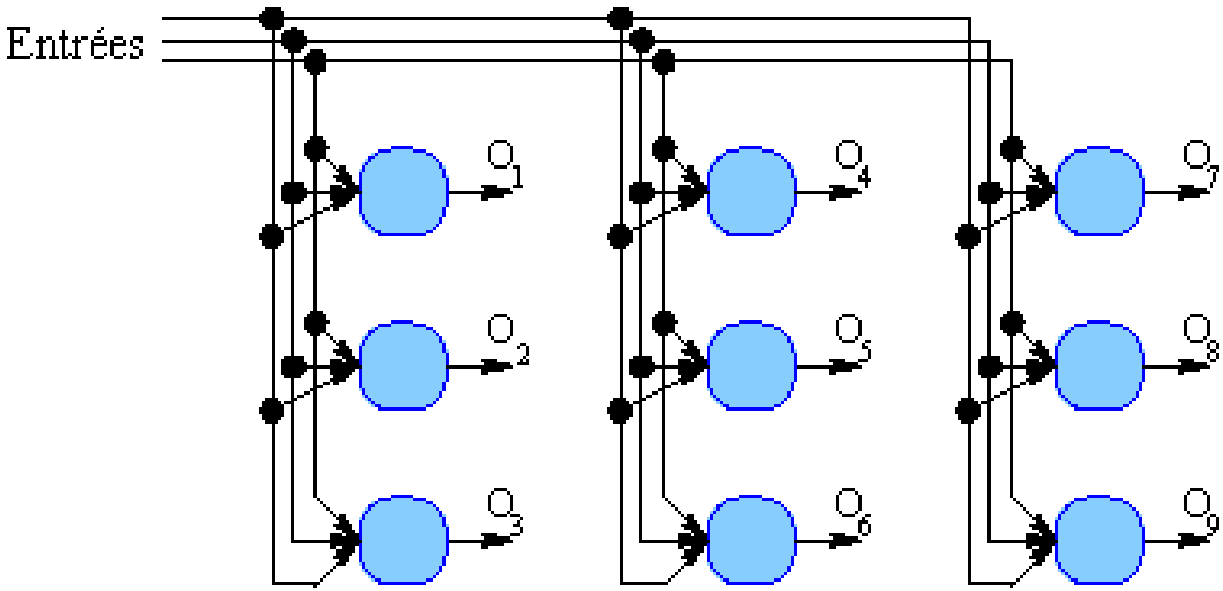
$O_i = 0$ sinon

Réseau récursif ou de Hopfield



Chaque unité i est connectée à chaque autre unité j par un poids w_{ij}.
Les poids sont supposés symétriques : $w_{ij} = w_{ji}$

Réseau de treillis



3×3 bi-dimensionnelle

Algorithme génétique

Quel est le dispositif de résolution de problèmes le plus puissant de l'univers ?

- **L'électricité** : qui permet de faire fonctionner les ordinateurs, les ampoules, etc.

  Edf.

- **L'amour** : qui est la solution à tous les maux

  Aphrodite.

- **Le cerveau humain** : qui inventa les précédents

  Descartes.

- **Le mécanisme de l'évolution** : qui créa le cerveau humain

  Darwin.

Théorie de l'évolution

Quand l'utiliser ?

- Espace de recherche est vaste.
- La meilleure solution n'est pas indispensable.
- Approche de résolution du problème n'est pas bien comprise.
- Le problème possède trop de paramètres à optimiser simultanément.
- Le problème est difficile à décrire mathématiquement.

Résolution	Evolution
Problème	Environnement
Solution	Individu
Qualité	Adaptation (*fitness*)

Méthodes incomplètes

Intérêt :

- Rapide
- Une solution

Problème :

- Incomplète

Technique :

- Génération aléatoire
- Correction locale (réparation)

Exemple - Les reines

- Placer les reines simplement : 20
- Heuristiques : < 2000
  - Colonnes les plus contraintes : 100
  - Lignes les plus contraintes : 500
  - Choix des colonnes milieu 1 et 3 tiers, bord pour centre
- Aléatoire : 1 million en 5mn

Exemple - 3-Sat

Énoncé :

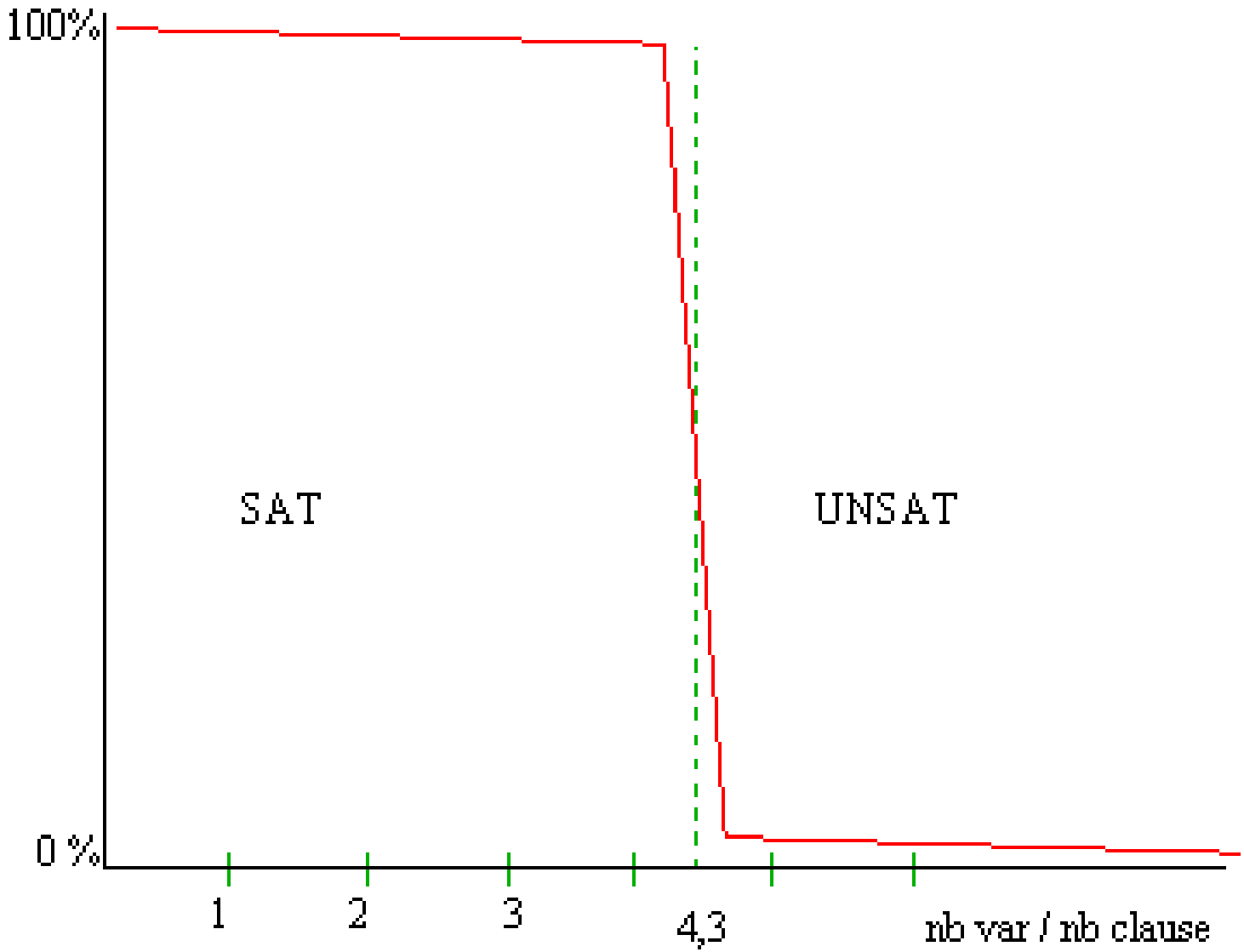
- Formule de la logique propositionnelle
- Longueur 3

Question :

- Le système est-il satisfiable ?

Exemple :

$$\begin{cases} a \vee b \vee c \\ \neg b \vee c \vee d \\ \neg a \vee b \vee d \end{cases}$$


100%
0 %
SAT
UNSAT
1
2
3
4,3
nb var / nb clause

Linguistique

Exemple :

- Y a-t-il un porte-monnaie dans la **pièce** voisine ?
  Si oui, y a-t-il des **pièces** dans ce porte-monnaie ?
- La belle ferme la porte.
- le dictateur promet aux manifestants de s'en aller
  le dictateur hurle aux manifestants de s'en aller
- le capitaine propose au second de partir en préretraite

Problème :

- Synonyme, homonyme (dictionnaire)
- Grammaire
- Contexte et Figure de style

Mécanisme

Analyse :

- Réécriture (hors contexte)
- Convergence (synonyme)
- Divergence (contexte)

Enchaînement :

- 1. Toto va à la banque
  2. Toto prend de l'argent
- 1 et 2 : retrait
- 2 et 1 : dépôt

Génération : `http://www.charabia.net`