# Extraction de motifs : Règles d'association et motifs séquentiels

**Maguelonne Teisseire**
TETIS – Irstea
teisseire@teledection.fr
http://www.lirmm. fr/~teisseire

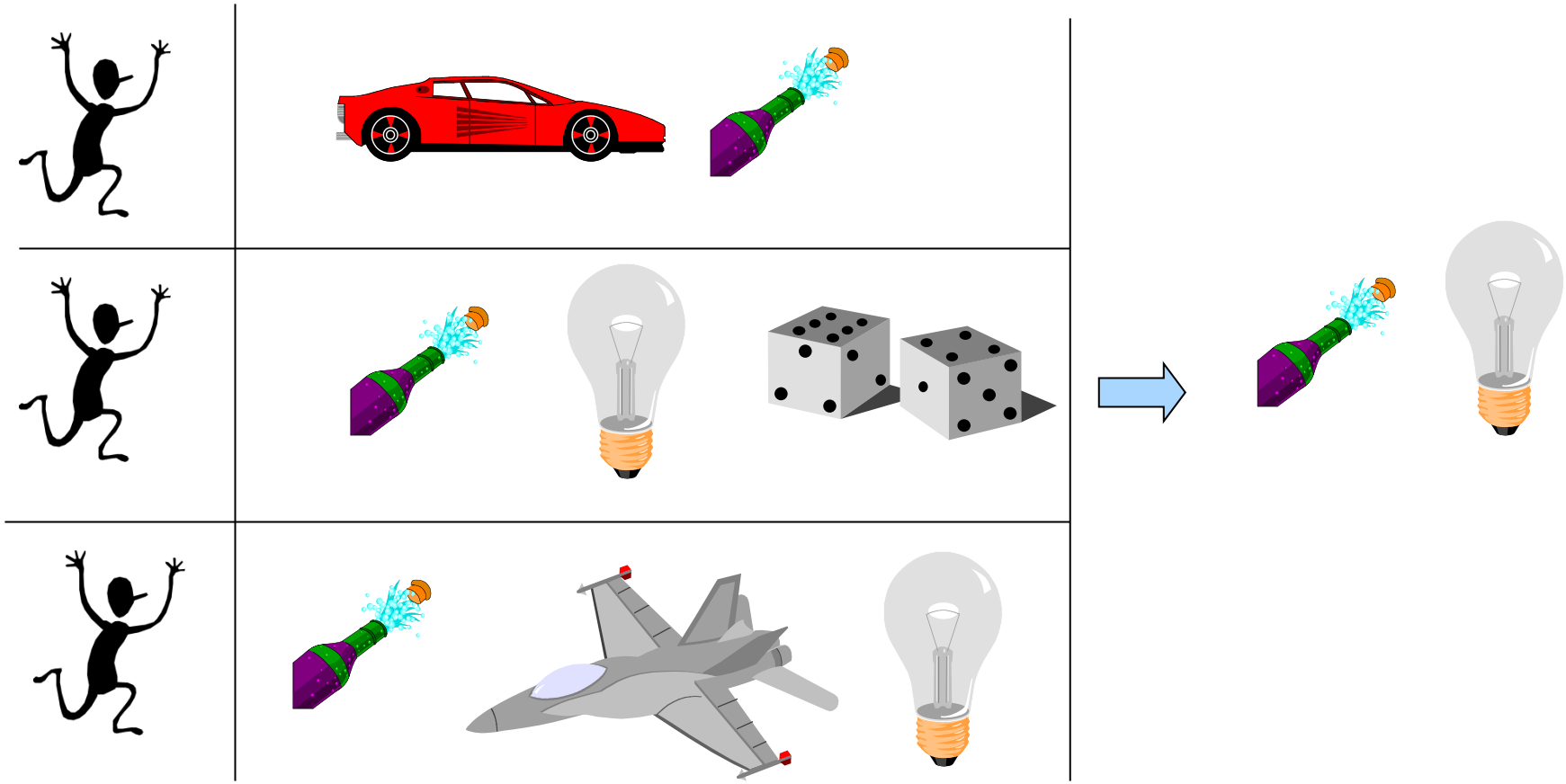
# Plan

- Règles d'association
- Motifs séquentiels
- Applications : Web Mining, Text Mining
- Conclusions

# « Panier de la ménagère »

- Recherche d'associations
  - recherche de corrélations entre attributs (items)
  - caractéristiques : « panier de la ménagère »
  - de très grandes données
  - limitations : données binaires
- Recherche de motifs séquentiels
  - recherche de corrélations entre attributs (items) mais en prenant en compte le temps entre items => comportement

# Recherche de règles d'association



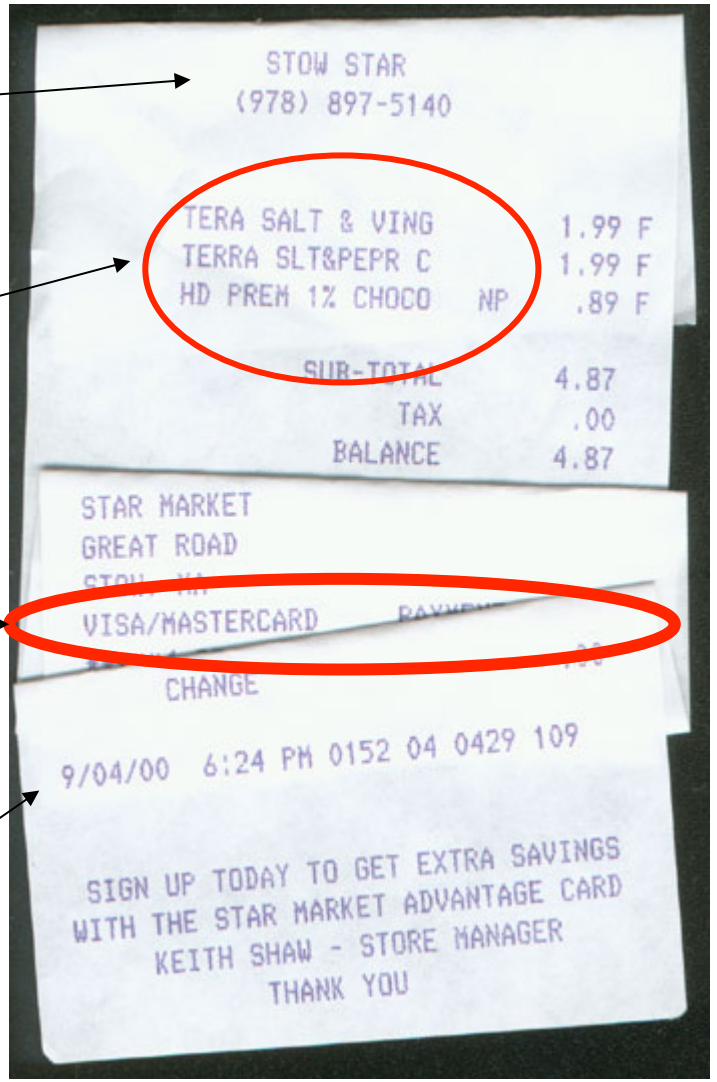
# Panier de la ménagère

Localisation

Produits achetés

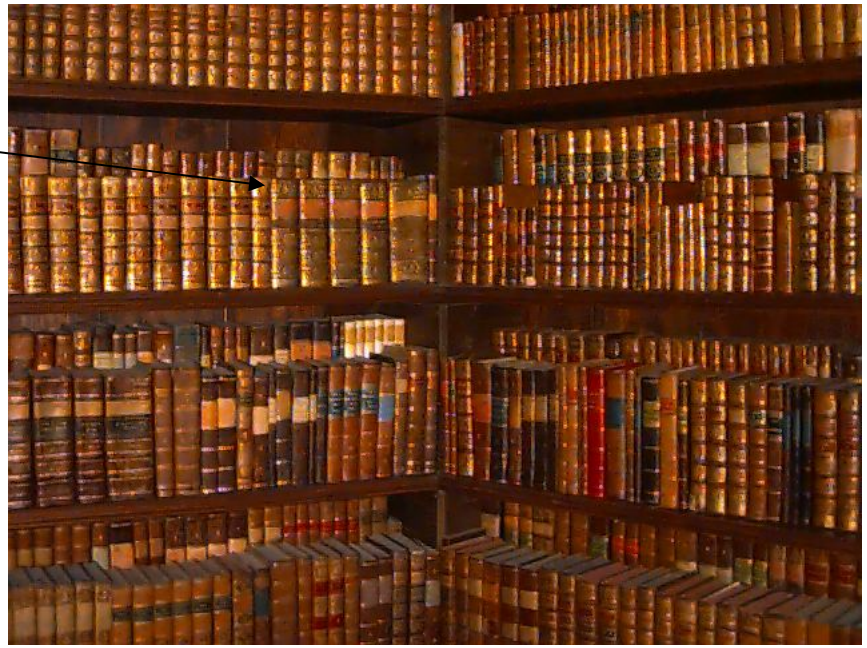
Identification

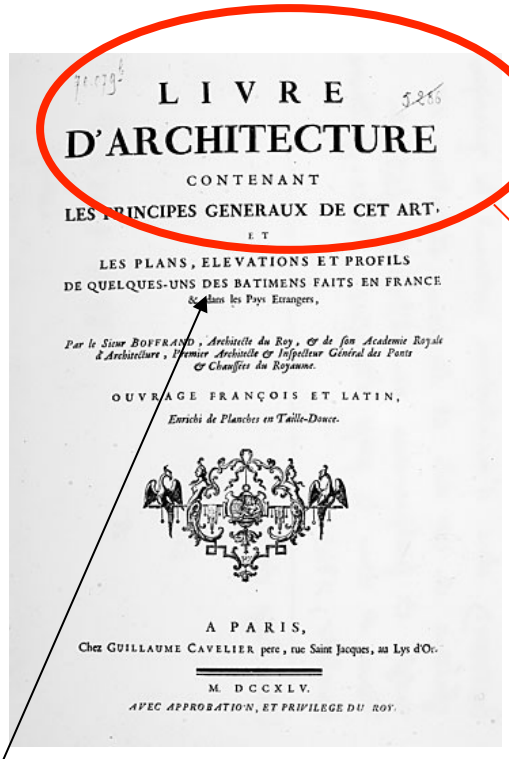
Date, heure



# Panier de la ménagère

Localisation





LIVRE
D'ARCHITECTURE
CONTENANT
LES PRINCIPES GENERAUX DE CET ART,
ET
LES PLANS, ELEVATIONS ET PROFILS
DE QUELQUES-UNS DES BATIMENS FAITS EN FRANCE
& dans les Pays Etrangers,

Par le Sieur BOFFRAND, Architecte du Roy, & de son Academie Royale d'Architecture, Premier Architecte & Inspecteur Général des Ponts & Chaussées du Royaume.

OUVRAGE FRANÇOIS ET LATIN,
Enrichi de Planches en Taille-Douce.

A PARIS,
Chez GUILLAUME CAVELIER pere, rue Saint Jacques, au Lys d'Or.

M. DCCXLV.
AVEC APPROBATION, ET PRIVILEGE DU ROY.

Premier paragraphe

« Livre d'architecture contenant les principes généraux … »

Identification

Position # Date

Mots # Produits

# Recherche de règles d'association

- Règles de la forme

**ANTECEDENT → CONSEQUENT [Support, Confiance]**

(support et confiance sont des mesures d'intérêt définies par l'utilisateur)

- Achat (x, « Beurre ») ET Achat (x, « Pain ») → Achat(x, « Lait ») [70%, 80%]
- Achat (x, « Bière ») ET Achat (x, « Gâteaux ») → Achat (x, « Couches ») [30%, 80%]
- Achat (x, « Caviar ») → Achat(x, « Champagne ») [10%, 90%]

# La légende

## Stories – Beer and Diapers

- Diapers and Beer. Most famous example of market basket analysis for the last few years. If you buy diapers, you tend to buy beer.
  - T. Blischok headed Terradata's Industry Consulting group.
  - K. Heath ran self joins in SQL (1990), trying to find two itemsets that have baby items, which are particularly profitable.
  - Found this pattern in their data of 50 stores/90 day period.
  - Unlikely to be significant, but it's a nice example that explains associations well.

*Ronny Kohavi ICML 1998*

# Interprétation

- R : X → Y (A%, B%)
  - ***Support*** *: portée de la règle*

    *Proportion de paniers contenant tous les attributs*

    *A% des clients ont acheté les 2 articles X et Y*

  - ***Confiance*** *:*

    *Proportion de paniers contenant le conséquent parmi ceux qui contiennent l'antécédent*

    *B% des clients qui ont acheté X ont aussi acheté Y*

  - Beurre, Pain → Lait [70%, 80%]
  - Bière, Gâteaux → Couches [30%, 80%]
  - Caviar → Champagne [10%, 90%]

# Utilisation des règles d'association

Bière, … →Couches

- **Couches** comme conséquent

  *déterminer ce qu'il faut faire pour augmenter les ventes*

- **Bière** comme antécédent

  *quel produit serait affecté si on n'arrête de vendre de la bière*

- **Bière** comme antécédent et **Couche** comme conséquent

  *quels produits devraient être vendus avec la Bière pour promouvoir la vente de couches*

# Définitions des ensembles fréquents

- Soit un ensemble I = {I1, I2, ..., Im} d'items, une transaction T est définie comme les sous-ensembles d'items dans I ($\subseteq$ I).
  - I = {Bière, Café, Couche, Gâteaux,Moutarde,Saucisse...}
  - T1 = {Café, Moutarde, Saucisse}
- Une transaction n'a pas de duplicats
- Soit une base de données D un ensemble de n transactions et chaque transaction est nommée par un identifiant (TID).
  - D = {{T1, {Café,Moutarde,Saucisse}},
    {T2, {Bière, Café, Gâteaux}}, ...}

# Une base de données

- Une représentation de la base de données D

| Client | Pizza | Lait | Sucre | Pommes | Café |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 | 0 | 0 |
| 3 | 1 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | 1 | 1 |

- En fait ….

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

# Définition des ensembles fréquents (cont.)

- Une transaction T **supporte** un ensemble X ⊆ I si elle contient tous les items de X (X ⊆ T).
  - T1 supporte {Café, Moutarde,Saucisse}
- Support de X (**Supp(X)**) : fraction de toutes les transactions dans D qui supportent X.
- Si supp(X) ≥ $s_{min}$ l'ensemble X est dit **fréquent**.

- Un ensemble d'items (*itemset*) X de cardinalité

k = |X| est appelé un *k-itemset*.

3-itemset : {Café, Moutarde, Saucisse}

# Propriétés des ensembles fréquents

- **Propriété 1** : *support pour les sous-ensembles*
  - Si A ⊆ B pour les itemsets A, B alors supp(A) >= supp(B) car toutes les transactions dans D qui supportent B supportent aussi nécessairement A.

  A={Café, Moutarde}, B ={Café, Moutarde, Saucisse}

- **Propriété 2** : *les sous-ensembles d'ensembles fréquents sont fréquents*

- **Propriété 3** : *les sur-ensembles d'ensembles non fréquents sont non fréquents* (anti-monotonie)

# Définition des Règles d'association

- Une règle d'association est une implication de la forme

$$R : X \rightarrow Y$$

où X et Y sont des itemsets disjoints :

$$X, Y \subseteq I \text{ et } X \cap Y = \varnothing.$$

Bière, Gâteaux → Couches

# Définition des Règles d'association (cont.)

- Confiance (*confidence*) dans une règle R
- Si une transaction supporte X, elle supporte aussi Y avec une certaine probabilité appelée **confiance** de la règle (conf(R)).

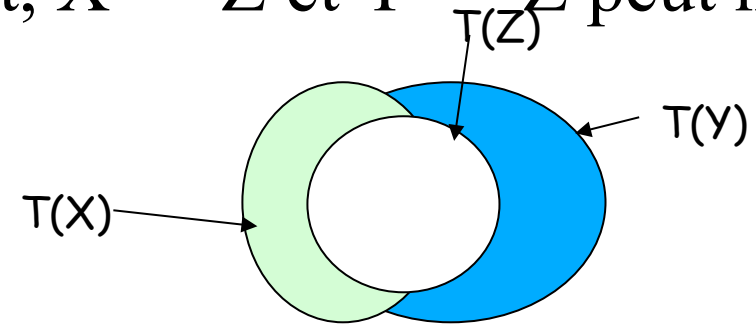
$$conf(R) = p(Y \subseteq T \mid X \subseteq T)$$
$$= p(Y \subseteq T \wedge X \subseteq T) / p(X \subseteq T)$$
$$= support(X \cup Y) / support(X)$$

$$conf(R) = \frac{Supp(Bière, Gâteaux, Couches)}{Supp(Bière, Gâteaux)} \geq confiance\ ?$$

# Propriétés des règles d'association

- **Propriété 4** : *pas de composition des règles*
  - Si X → Z et Y → Z sont vrais dans D, X U Y → Z n'est pas nécessairement vrai.
  - Considérons le cas où X ∩ Y = Ø et les transactions dans D supportent Z si et seulement si elles supportent X ou Y, alors l 'ensemble X U Y a un support de 0 et donc X U Y → Z a une confiance de O%.

- **Propriété 5** : *décomposition des règles*
  - Si X U Y → Z convient, X → Z et Y → Z peut ne pas être vrai.



# Propriétés des règles d'association

- **Propriété 6** : *pas de transitivité*
  - Si X → Y et Y → Z , nous ne pouvons pas en déduire que X → Z.

- **Propriété 7** : *déduire si une règle convient*
  - Si A → (L-A) ne vérifie pas la confiance alors nous n'avons pas B → (L-B) pour les itemsets L, A, B et B ⊆ A.

# En résumé

- Itemsets : A, B  ou B, E, F
- Support pour un itemset

  Supp (A,D)=1

  Supp (A,C) = 2
- Itemsets fréquents (minSupp=50%)

  {A,C} est un itemset fréquent
- Pour minSupp = 50% et minConf = 50%, nous avons les règles suivantes :

  A → C [50%, 50%]

  C → A [50%, 100%]

| Trans. ID | Items |
|---|---|
| 1 | A, D |
| 2 | A, C |
| 3 | A, B, C |
| 4 | A, B, E, F |

# Schéma algorithmique de base

- La plupart des approches utilise le même schéma algorithmique
- Pour construire les règles d ’association, le support de tous les itemsets fréquents dans la base doit être calculé
- L ’algorithme procède en deux phases :
  - 1) Génération de tous les ensembles fréquents
  - 2) Génération des règles d ’association

# Comptage des itemsets

- Une première approche
- I = {A, B,C}
- Génération de tous les cas possibles :

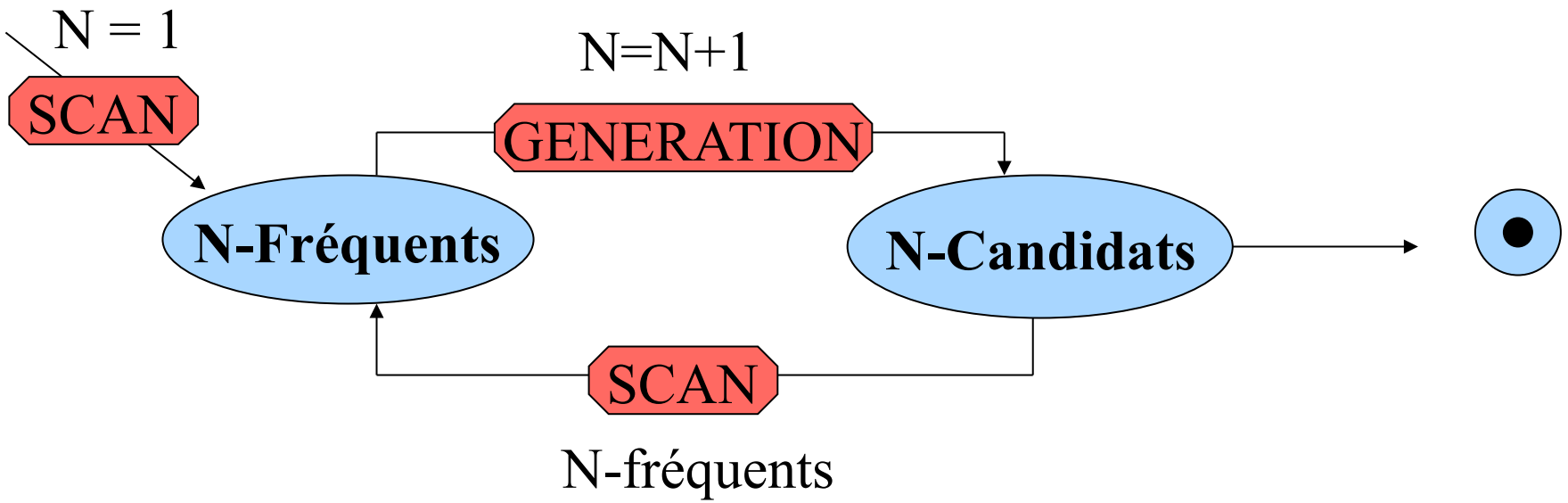
  {∅},{A}, {B}, {C},
  {A,B}, {A,C}, {B,C}
  {A,B,C}

- Comptage du support

# Génération des ensembles fréquents

- Le nombre d ’ensemble fréquent potentiel est égal à la taille du produit cartésien de tous les items …. qui croit exponentiellement en fonction du nombre d ’items considérés.
- Approche naïve : recherche exhaustive et test de tous les ensemble du produit cartésien pour savoir s ’ils sont fréquents
- 1000 items => $2^{1000}$ ensembles à considérer

# Vers un algorithme générique



# Construction des règles

- Pour chaque ensemble fréquent X, chaque sous-ensemble est choisi comme antécédent de la règle, le reste devenant la partie conséquent.
- Comme X est fréquent, tous les sous-ensembles sont fréquents (Propriété 3) donc leur support est connu. La confiance d'une règle est calculée et une règle est conservée ou pas selon la confiance minimale.
- Amélioration : (Propriété 7) quand une règle échoue, aucun sous ensembles de l 'antécédent n 'est à considérer.

# Bref historique

- Problématique initiée en 1993
- CPU vs. I/O
- De nombreux algorithmes ...

AIS - *R. Agrawal, T. Imielinski and A. Swami - ACM SIGMOD 1993*

SETM - *Houtsma and Swami - IBM Technical Record*

APRIORI - *R. Agrawal and R. Srikant - VLDB 1994*

PARTITION - *A. Sarasere, E. Omiecinsky and S. Navathe - VLDB 1995*

SAMPLING - *H. Toivonen - VLDB 1996*

DIC - *S. Brin, R. Motwani, J.Ulman and S. Tsur - ACM SIGMOD 1997*

PrefixSpan - *J. Pei, J. Han, …. - ICDE'01*

SPADE - *M. Zaki - Machine Learning'01*

….2006, 2007

# L'algorithme APRIORI

- But : minimiser les candidats
- Principe : générer seulement les candidats pour lesquels tous les sous-ensembles ont été déterminés fréquents

- Génération des candidats réalisée avant et de manière séparée de l'étape de comptage

# L'algorithme APRIORI

*Input* : $C_k$: itemsets candidats de taille k

*Output* : $L_k$ : itemsets fréquents de taille k

$L_1$ = {items fréquents};

for (k = 1; $L_k$ !=∅; k++) do

$C_{k+1}$ = candidats générés à partir de $L_k$;

Pour chaque transaction t de la base de données, incrémenter le compteur de tous les candidats dans $C_{k+1}$ qui sont contenus dans t

$L_{k+1}$ = candidats dans $C_{k+1}$ avec minSupp

return $\cup_k L_k$;

# Détails d'APRIORI

- Comment générer les candidats ?
  - Etape 1: auto-jointure sur $L_k$
  - Etape 2: élagage
- Comment compter le support des candidats ?

# Génération des candidats

- Les items de $L_{k-1}$ sont ordonnés par ordre lexicographique

- Etape 1: auto-jointure sur $L_{k-1}$

  INSERT INTO $C_k$

  SELECT $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

  FROM $L_{k-1}\ p, L_{k-1}\ q$

  WHERE $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$

- Etape 2: élagage

  For each itemset $c$ in $C_k$ do

  For each ($k$-1)-subsets $s$ of $c$ do if ($s$ is not in $L_{k-1}$) then delete $c$ from $C_k$

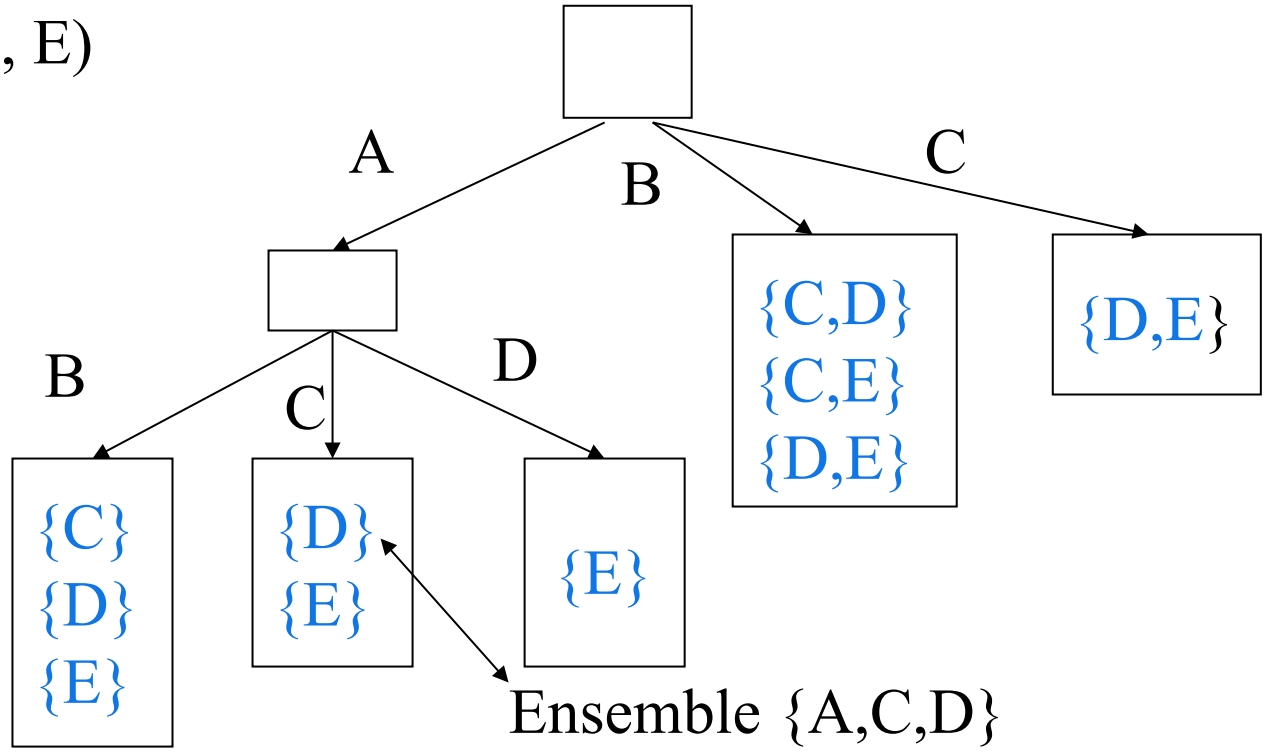
# Génération des candidats : exemple

- $L_3$={*abc, abd, acd, ace, bcd*}
- Auto-jointure : $L_3$*$L_3$
  - *abcd* à partir de *abc* et *abd*
  - *acde* à partir de *acd* et *ace*
- Élagage :
  - *acde* est supprimé car *ade* n'est pas dans $L_3$
- $C_4$={*abcd*}

# Stockage des candidats

- un arbre (structure de hash-tree)

structure de tous les 3-candidats possibles pour 5 items (A, B, C, D, E)



# Comptage du support des candidats

- Parcourir la base. Pour chaque tuple extrait t, compter tous les candidats inclus dedans
  - Rechercher toutes les feuilles qui peuvent contenir les candidats
  - Hachage sur chaque item du tuple et descente dans l ’arbre des candidats
- Dans les feuilles de l’arbre vérifier ceux effectivement supportés par t
- Incrémenter leur support

# Illustration

| CID | Items |
|---|---|
| 1 | A B |
| 2 | A B C D E F |
| 3 | B D G |
| 4 | B E G |
| 5 | D F G |
| 6 | DEG |
| 7 | B E |
| 8 | B D E F |

Support minimal = 1

# Illustration

| C1 | Support |
|---|---|
| A | 2 |
| B | 6 |
| C | 1 |
| D | 5 |
| E | 5 |
| F | 3 |
| G | 4 |

L1 = {{A},{B},{C},{D},{E},{F},{G}} 1-itemsets fréquents

# Illustration

| C2 | Support | C2 | Support |
|---|---|---|---|
| AB | 2 | CD | 1 |
| AC | 1 | CE | 1 |
| AD | 1 | CF | 1 |
| AE | 1 | **CG** | **0** |
| AF | 1 | DE | 3 |
| **AG** | **0** | DF | 3 |
| BC | 1 | DG | 3 |
| BD | 3 | EF | 2 |
| BE | 4 | EG | 2 |
| BF | 2 | FG | 1 |
| BG | 2 | | |

2-itemsets fréquents{{A,B},{A,C},{A,D},{A,E},{A,F},{B,C},{B,D},{B,E},
{B,F},{B,G}, {C,D},{C,E},{C,F},{D,E},{D,F},{D,G},{E,F},{E,G},{F,G}}

# Illustration

| C3 | Support | C3 | Support |
|---|---|---|---|
| ABC | 1 | BDE | 2 |
| ABD | 1 | BDF | 2 |
| ABE | 1 | BDG | 1 |
| ABF | 1 | BEF | 2 |
| ACD | 1 | BEG | 1 |
| ACE | 1 | **BFG** | **0** |
| … | … | … | … |
| BCF | 1 | **EFG** | **0** |

L3 = {{A,B,C},{A,B,D},{A,B,E},{A,B,F},{A,C,D}, … {D,F,G}}

{B,C,G} élagué par Apriori-Gen car {C, G} n ’appartient pas à L2

# Illustration

| C4 | Support | C4 | Support |
|---|---|---|---|
| ABCD | 1 | ACEF | 1 |
| ABCE | 1 | ADEF | 1 |
| ABCF | 1 | BCDE | 1 |
| ABDE | 1 | BCDF | 1 |
| ABDF | 1 | BCEF | 1 |
| ABEF | 1 | BDEF | 2 |
| ACDE | 1 | **BDEG** | **0** |
| ACDF | 1 | CDEF | 0 |

L4 = {{A,B,C,D},{A,B,C,E},{A,B,C,F}, … {C,D,E,F}}

{B,D,F,G}, {B,E,F,G} élagués car {B,F,G} n ’appartient pas à L3

{D,E,F,G} élagué car {E,F,G} n ’appartient pas à L3

# Illustration

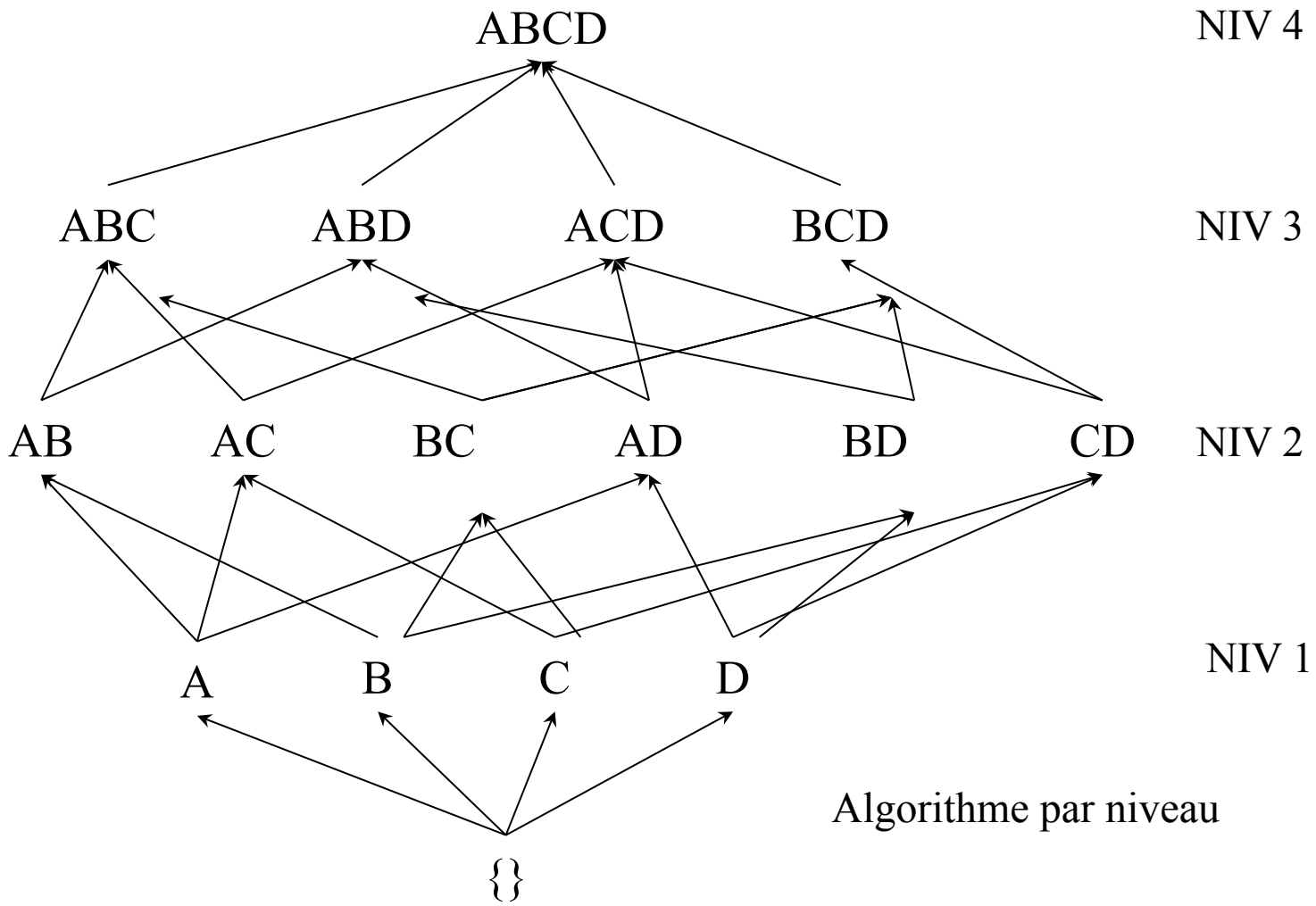
| C6 | Support |
|---|---|
| ABCDE | 1 |

6-itemsets fréquents  L6 = {{A,B,C,D,E,F}}

C7 = {∅} => l 'algorithme se termine.

7 balayages pour déterminer tous les itemsets fréquents

# Espace de recherche




Algorithme par niveau

# Quelques conclusions

- De nombreux travaux
  - De nouvelles approches condensées
  - De nouvelles contraintes (réduire l'espace de recherche)
  - Préservation de la vie privée
  - Approches Incrémentales
  - Règles plus générales
  - Définir de nouvelles mesures (lift, implication, …)

# Règles d'association incrémentales

- Générer les règles dans une base dynamique
- Problème : les algorithmes considèrent des bases statiques
- Objectifs :
  - Chercher les itemsets fréquents dans D
  - Chercher les itemsets fréquents dans D $\cup$ {$\Delta$ D}
- Doit être fréquent dans D ou $\Delta$ D
- Sauvegarder tous les fréquents, la bordure
- … Data Streams (Flots de Données)

# Des règles plus générales

- Les règles négatives

  Expr(Ci) → Expr(Cj) avec AND, OR, NOT

- Les règles sur plusieurs dimensions
- Les règles à attributs variables

  Age ∈[x,y] => Salaire > 45 K€ (5%; 30%)

- Les règles approximatives
- Les règles avec généralisation

  Associée à une taxonomie

# Utilité des règles

- La règle utile contenant des informations de qualité qui peuvent être mises en pratique

  ex : *le samedi, les clients des épiceries achètent en même temps de la bière et des couches*

- Résultats connus par quiconque

  ex : *les client des épiceries achètent en même temps du pain et du beurre*

- Résultats inexplicables difficiles à situer et donc à expliquer

  ex : *lors de l'ouverture d'une quincaillerie, parmi les articles les plus vendus on trouve les abattants de toilette*

# D'autres mesures

| Articles | A | B | C | A, B | A, C | B, C | A, B, C |
|---|---|---|---|---|---|---|---|
| Fréquences (%) | 45 | 42,5 | 40 | 25 | 20 | 15 | 5 |

- Si on considère les règles à trois articles, elles ont le même support 5%. Le niveau de confiance est alors :

| Règle | Confiance |
|---|---|
| A, B → C | 0,20 |
| A, C → B | 0,25 |
| B,C → A | 0,33 |

- La règle « B, C → A » possède la plus grande confiance. si B et C apparaissent simultanément dans un achat alors A y apparaît aussi avec une probabilité estimée de 33%.

# D’autres mesures (cont.)

| **Articles** | A | B | C | A, B | A, C | B, C | A, B, C |
|---|---|---|---|---|---|---|---|
| **Fréquences (%)** | **45** | 42,5 | 40 | 25 | 20 | 15 | 5 |

- A apparaît dans 45% des achats. Il vaut donc mieux prédire A sans autre information que de prédire A lorsque B et C apparaissent.
- l'*amélioration* permet de comparer le résultat de la prédiction en utilisant la fréquence du résultat

$$Amélioration = confiance / frequence(résultat)$$

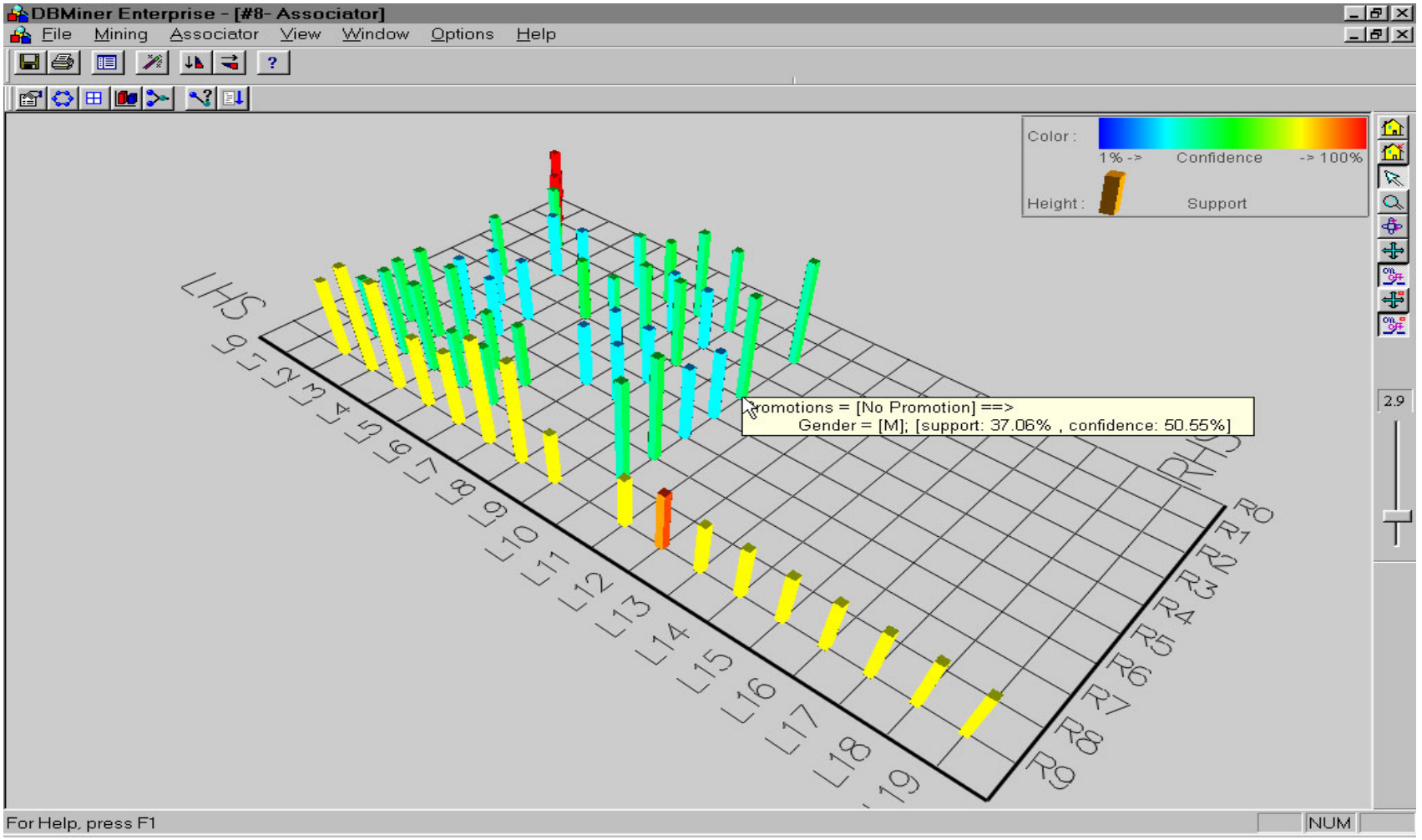
# D’autres mesures (cont.)

- Une règle est intéressante lorsque l'amélioration est supérieure à 1. Pour les règles choisies, on trouve :

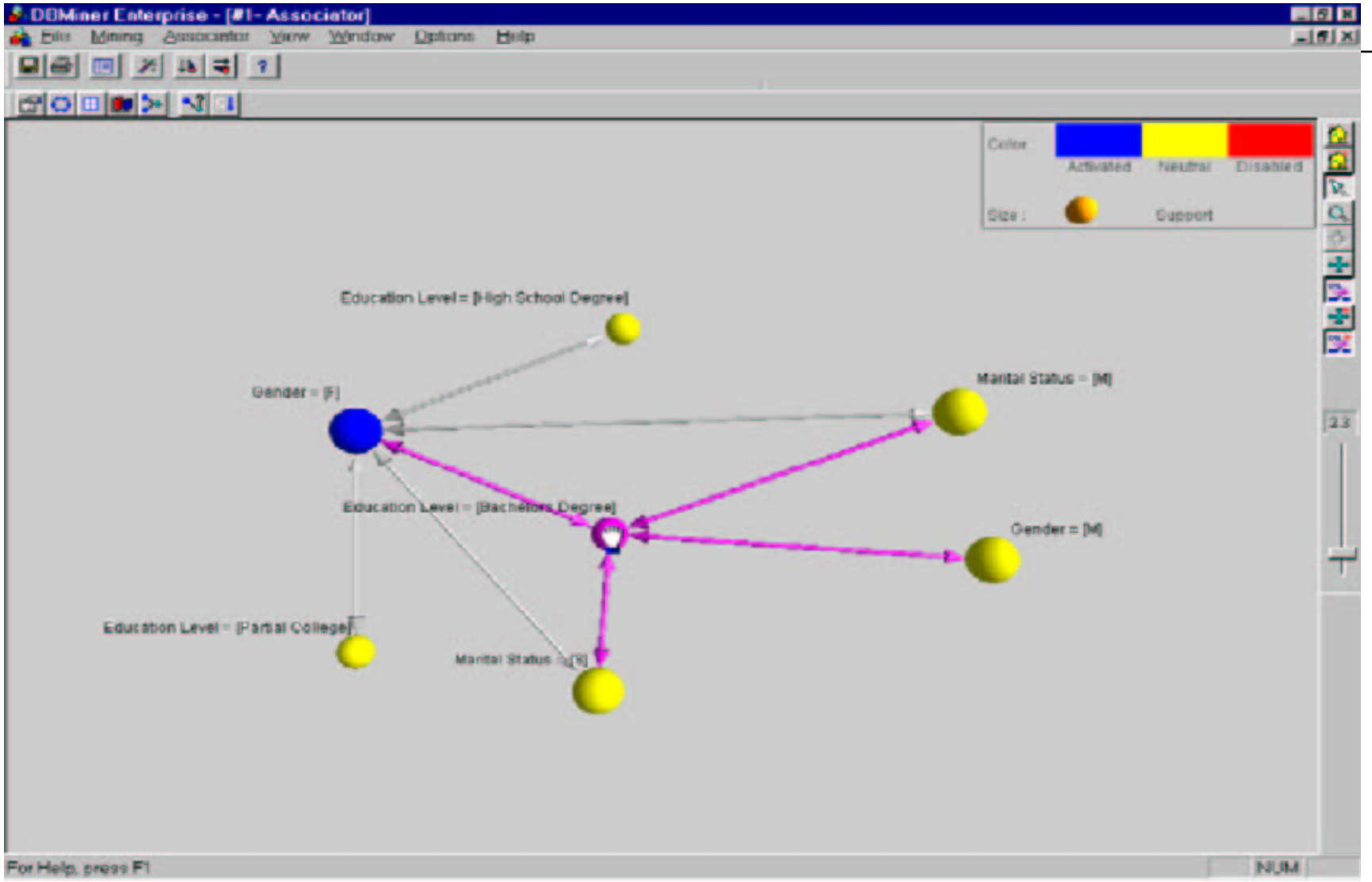
| Règle | Confiance | Freq(résultat) | Amélioration |
|---|---|---|---|
| A, B→ C | 0.20 | 40% | 0.50 |
| A,C → B | 0.25 | 42.5% | 0.59 |
| B,C → A | 0.33 | 45% | 0.74 |

- Par contre, la règle si « A → B » possède un support de 25%, une confiance de 0.55 et une amélioration de 1.31, cette règle est donc la meilleure.
- En règle générale, la meilleure règle est celle qui contient le moins d'articles.

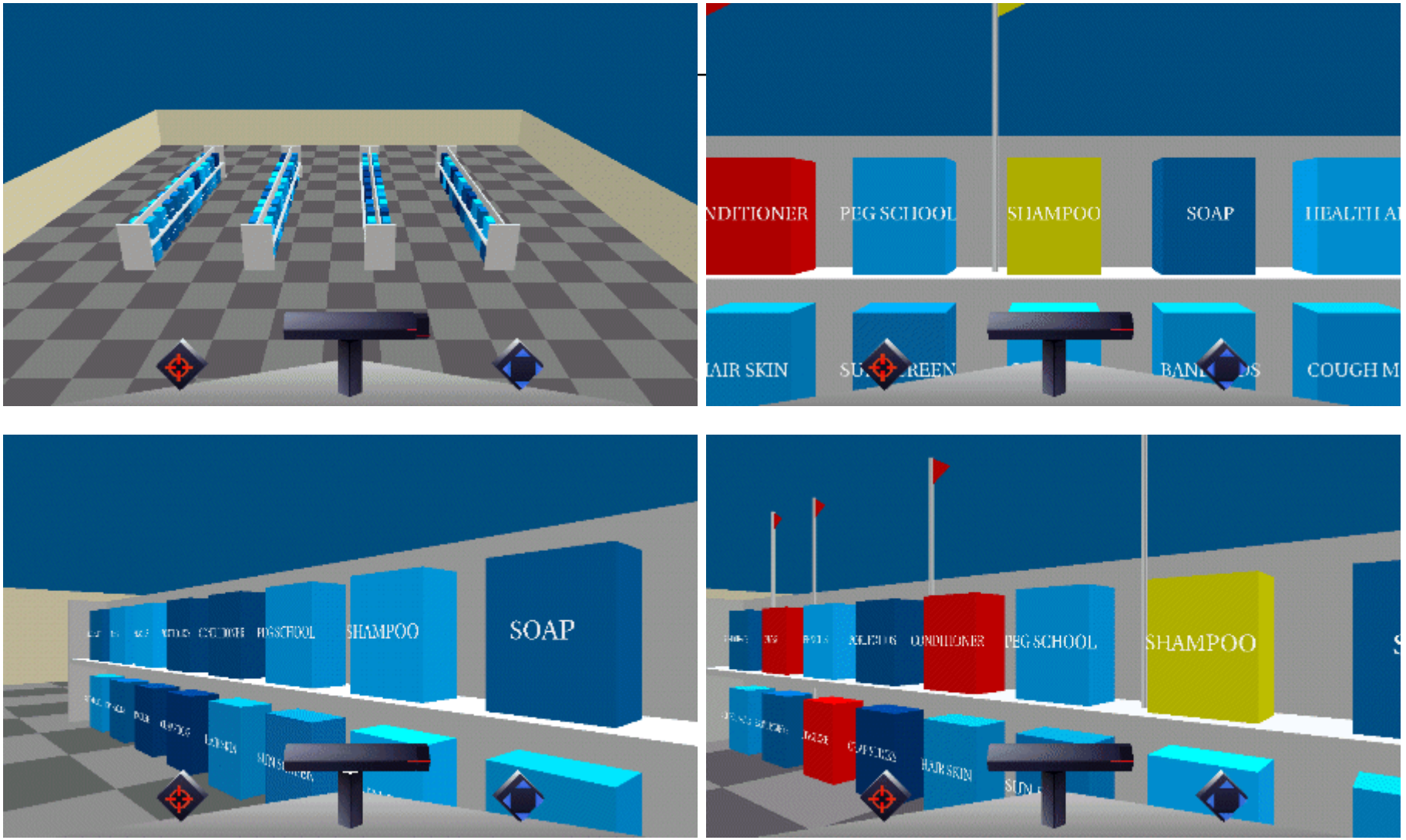
# Visualisation




DBMiner (www.dbminer.com)




DBMiner (www.dbminer.com)




Intelligent Miner (www.ibm.com)

# Plan

- Contexte général
- Règles d'association
- Motifs séquentiels
- Applications : Web Mining, Text Mining
- Conclusions

# Pourquoi la recherche de séquence ?

- Un important domaine de recherche pour le data mining avec de très nombreuses applications
  - Analyse des achats des clients
  - Analyse de puces ADN
  - Processus
  - Conséquences de catastrophes naturelles
  - Web mining
  - Détection de tendances dans des données textuelles

# Recherche de Motifs Séquentiels

- Même problématique mais avec le temps
- Item : « un article »
- Transaction : un client + un itemset + une estampille temporelle $T = [C, (a,b,c)_5]$
- Séquence : liste ordonnée d'itemsets
- Séquence de données : « activité du client »

  Soit $T_1$, $T_2$, … $T_n$, les transactions du client C, la séquence de données de C est :

  $[C, <itemset(T_1)\ itemset(T_2)\ \ldots.\ itemset(T_n)>]$

# Recherche de Motifs Séquentiels

- Support minimal : nombre minimum d'occurrences d'un motif séquentiel pour être considéré comme fréquent

- Attention l'occurrence n'est prise en compte qu'une fois dans la séquence

Support (20) dans <(10) (20 30) (40) (20)>=1

# Inclusion

- Inclusion : Soient $S_1 = <a_1\ a_2\ \ldots\ a_n>$ et $S_2 = <b_1\ b_2\ \ldots\ b_n>$ $S_1 \subseteq S_2$ ssi

  $i_1 < i_2 < \ldots < i_n$ / $a_1 \subseteq b_{i1}, \ldots. a_n \subseteq b_{in}$

- S1 = <(10) (20 30) (40) (20)>

  S2 = <(20) (40)> $\subseteq$ S1

  S3 = <(20) (30)> n'est pas incluse dans S1

# Problématique

- Soit D une base de données de transactions de clients. Soit σ une valeur de support minimal

  Rechercher toutes les séquences S telles que :

  support(S) ≥σ dans D

- 50% des personnes qui achètent du vin et du fromage le lundi achètent aussi du pain le vendredi

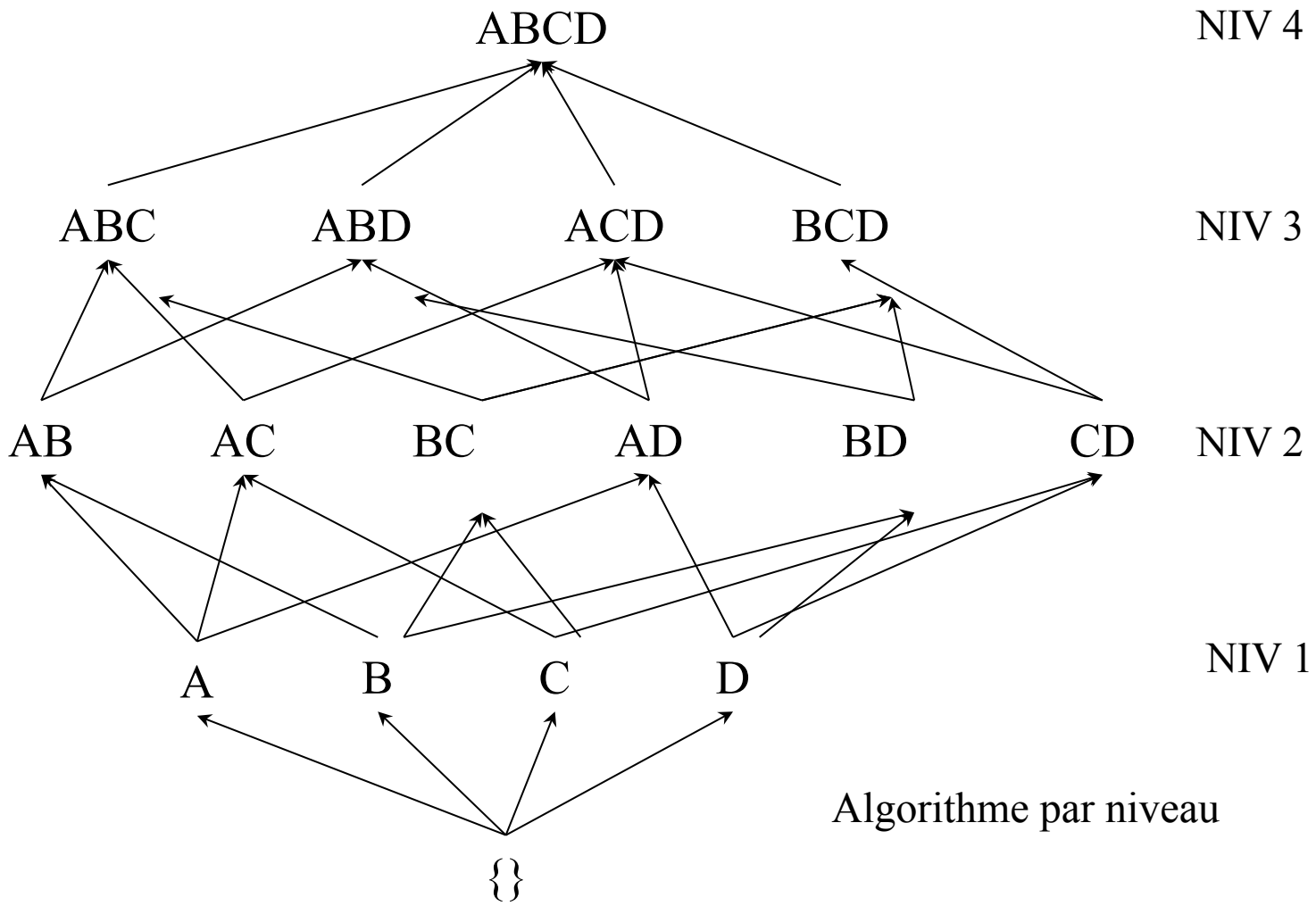
  <(French wine, cheese) (bread)>

# Illustration

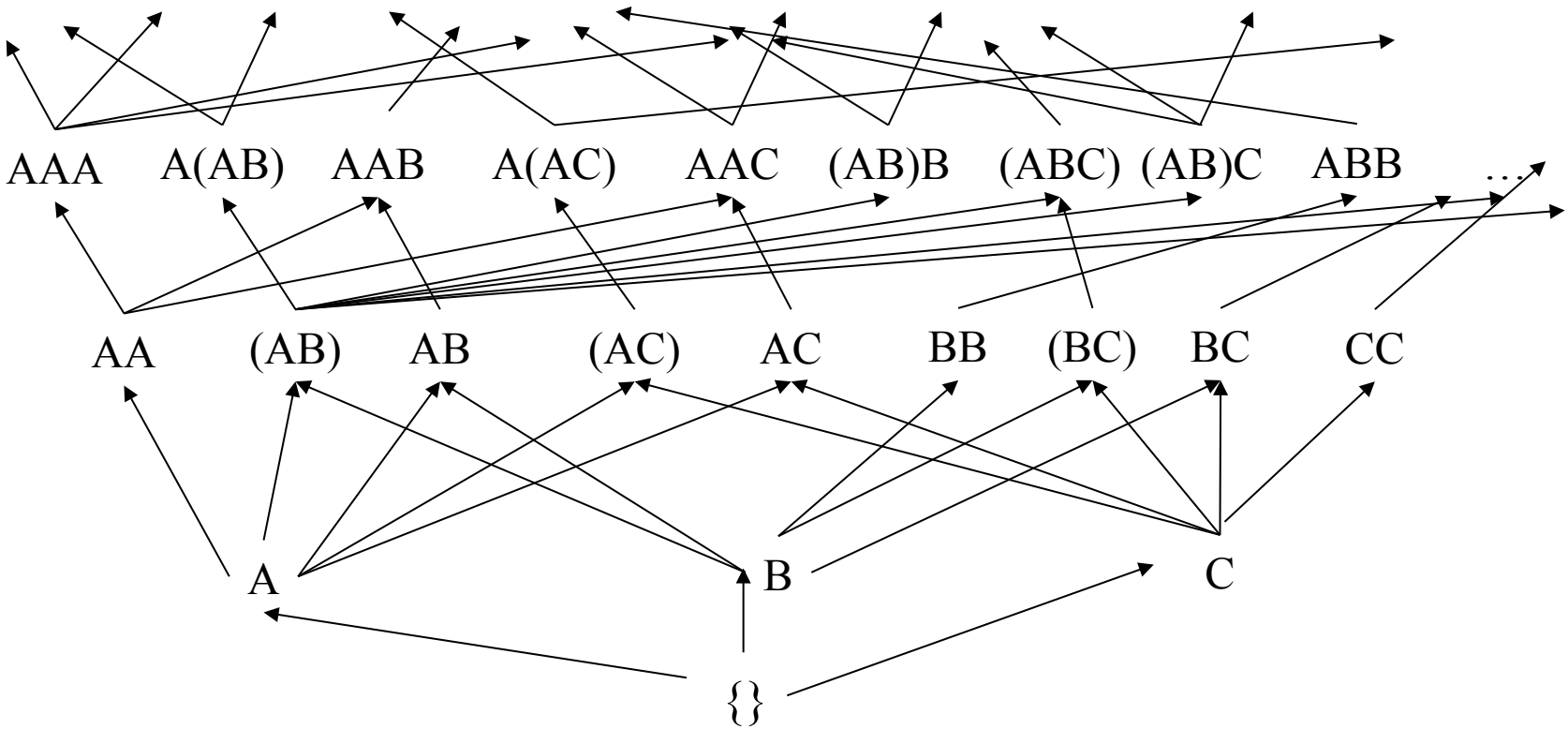
| Clients | Date1 | Date2 | Date3 | Date4 |
|---|---|---|---|---|
| C1 | 10 20 30 | 20 40 50 | 10 20 60 | 10 40 |
| C2 | 10 20 30 | 10 20 30 | | 20 30 60 |
| C3 | 20 30 50 | | 10 40 60 | 10 20 30 |
| C4 | 10 30 60 | 20 40 | 10 20 60 | 50 |

Support = 60% (3 clients) => <(10 30) (20) (20 60)>

# Itemsets :Espace de recherche



# Motifs Séquentiels : l'espace de recherche



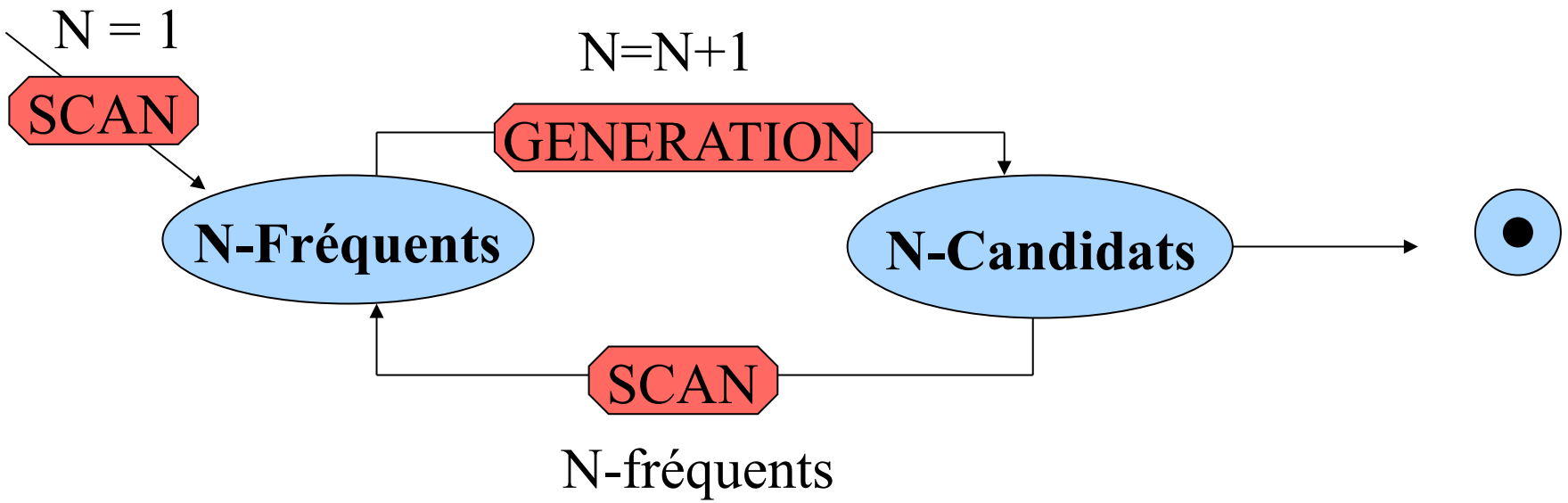
# La propriété d'antimonotonie

- Une propriété essentielle (c.f. Apriori [AIS93])
  - Si une séquence n'est pas fréquente, aucune des super-séquences de S n'est fréquente!

Support (<(10) (20 30)>) < minsupp

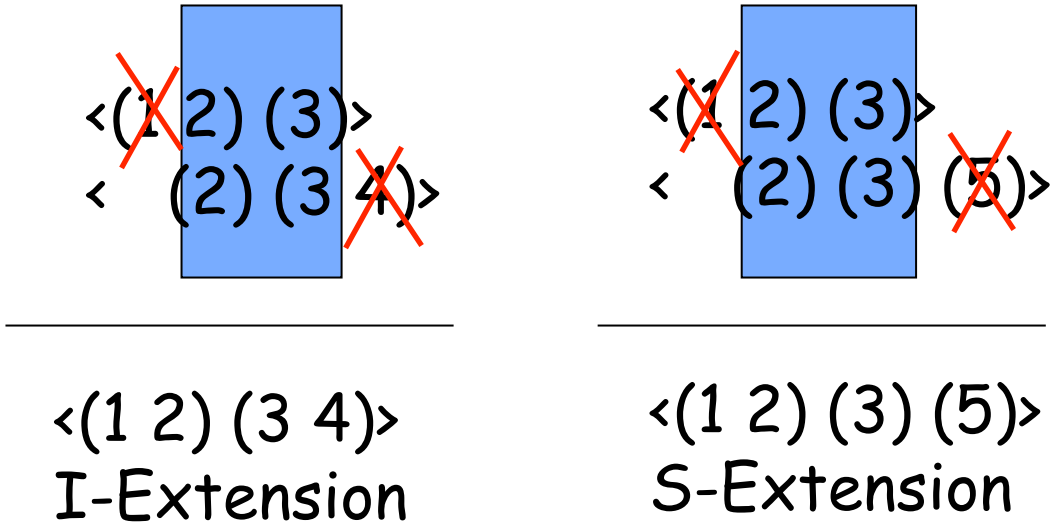
Support (<(10) (20 30) (40)>)<< minsupp

# Vers un algorithme générique



# Génération des candidats

- S-Extension : ajout d'une séquence
- I-Extension : ajout d'un itemset



# GSP

- A la APRIORI [Srikant, Agrawal, EDBT'96]

L=1

While ($Result_L$ != NULL)

    Candidate Generate

    Prune

    Test

    L=L+1

# Recherche des séquences de taille 1

- Candidats initiaux : toutes les séquences réduites à un item
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Un passage sur la base pour compter le support des candidats

| Seq. ID | Séquence |
|---|---|
| 10 | <(bd)cb(ac)> |
| 20 | <(bf)(ce)b(fg)> |
| 30 | <(ah)(bf)abf> |
| 40 | <(be)(ce)d> |
| 50 | <a(bd)bcb(ade)> |

*minSupp* =2

| Cand | Sup |
|---|---|
| **<a>** | **3** |
| **<b>** | **5** |
| **<c>** | **4** |
| **<d>** | **3** |
| **<e>** | **3** |
| **<f>** | **2** |
| ~~<g>~~ | 1 |
| ~~<h>~~ | 1 |

# Le Processus

**5th scan** : 1 candidate
1 length-5 seq pattern

<(bd)cba>

**4th scan** : 8 candidates
6 length-4 seq pat

<abba> <(bd)bc> …

**3rd scan** : 46 candidates
19 length-3 seq pat.

<abb> <aab> <aba> <baa> <bab> …

**2nd scan** : 51 candidates
19 length-2 seq pat.

<aa> <ab> … <af> <ba> <bb> … <ff> <(ab)> … <(ef)>

**1st scan** : 8 candidates
6 length-1 seq pattern

<a> <b> <c> <d> <e> <f> <g> <h>

# Génération des candidats de taille 2

S-Extension

| | <a> | <b> | <c> | <d> | <e> | <f> |
|---|---|---|---|---|---|---|
| <a> | <aa> | <ab> | <ac> | <ad> | <ae> | <af> |
| <b> | <ba> | <bb> | <bc> | <bd> | <be> | <bf> |
| <c> | <ca> | <cb> | <cc> | <cd> | <ce> | <cf> |
| <d> | <da> | <db> | <dc> | <dd> | <de> | <df> |
| <e> | <ea> | <eb> | <ec> | <ed> | <ee> | <ef> |
| <f> | <fa> | <fb> | <fc> | <fd> | <fe> | <ff> |

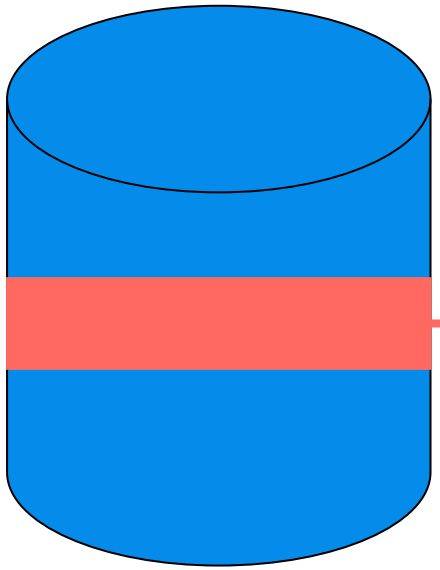
51 2-Candidats

I-Extension

| | <a> | <b> | <c> | <d> | <e> | <f> |
|---|---|---|---|---|---|---|
| <a> | | <(ab)> | <(ac)> | <(ad)> | <(ae)> | <(af)> |
| <b> | | | <(bc)> | <(bd)> | <(be)> | <(bf)> |
| <c> | | | | <(cd)> | <(ce)> | <(cf)> |
| <d> | | | | | <(de)> | <(df)> |
| <e> | | | | | | <(ef)> |
| <f> | | | | | | |

Sans la propriété d'anti-monotonie
8*8+8*7/2=92 candidats

# Comptage des supports des candidats



Un tuple
T={cid, <(a) (bc) (d)>}

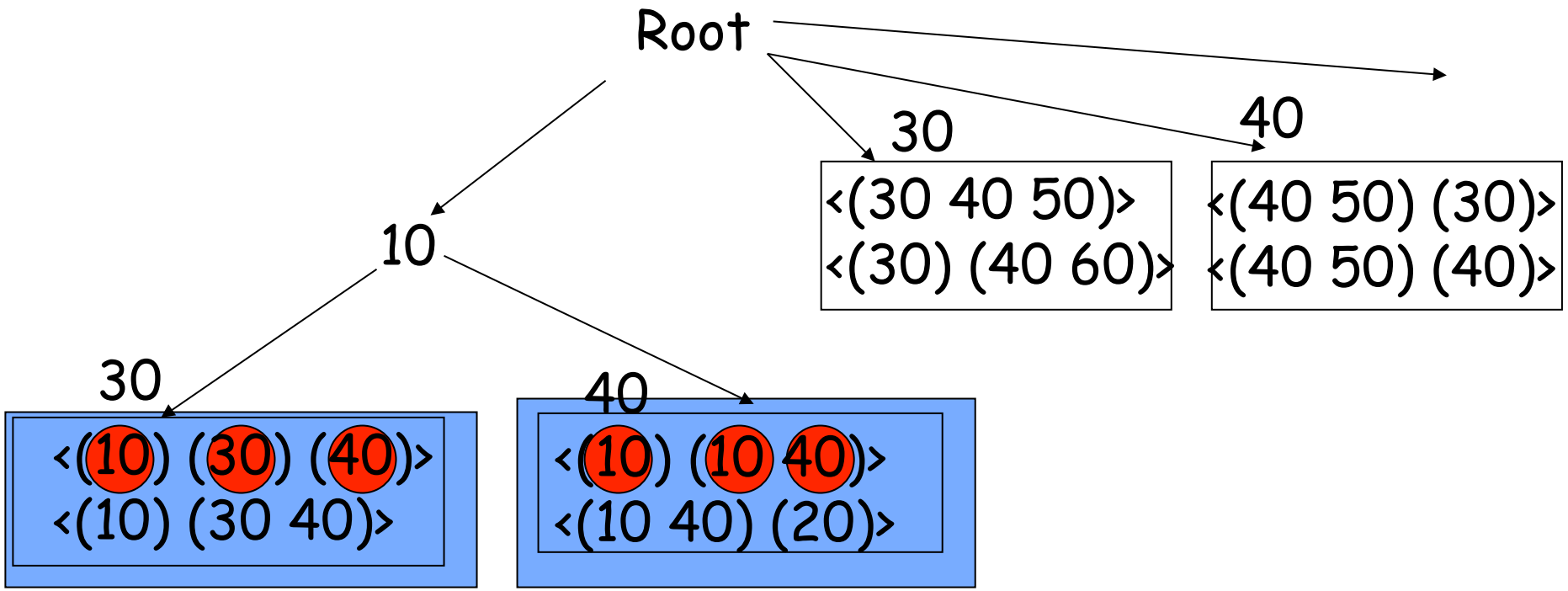
BASE DE DONNEES

Candidats
Stockés en
Mémoire centrale

| | |
|---|---|
| C1 | C1 ⊆ T ? |
| C2 | C2 ⊆ T ? |
| C3 | C3 ⊆ T ? |
| C4 | C4 ⊆ T ? |
| C5 | C5 ⊆ T ? |
| Cn | Cn ⊆ T ? |

MEMOIRE CENTRALE

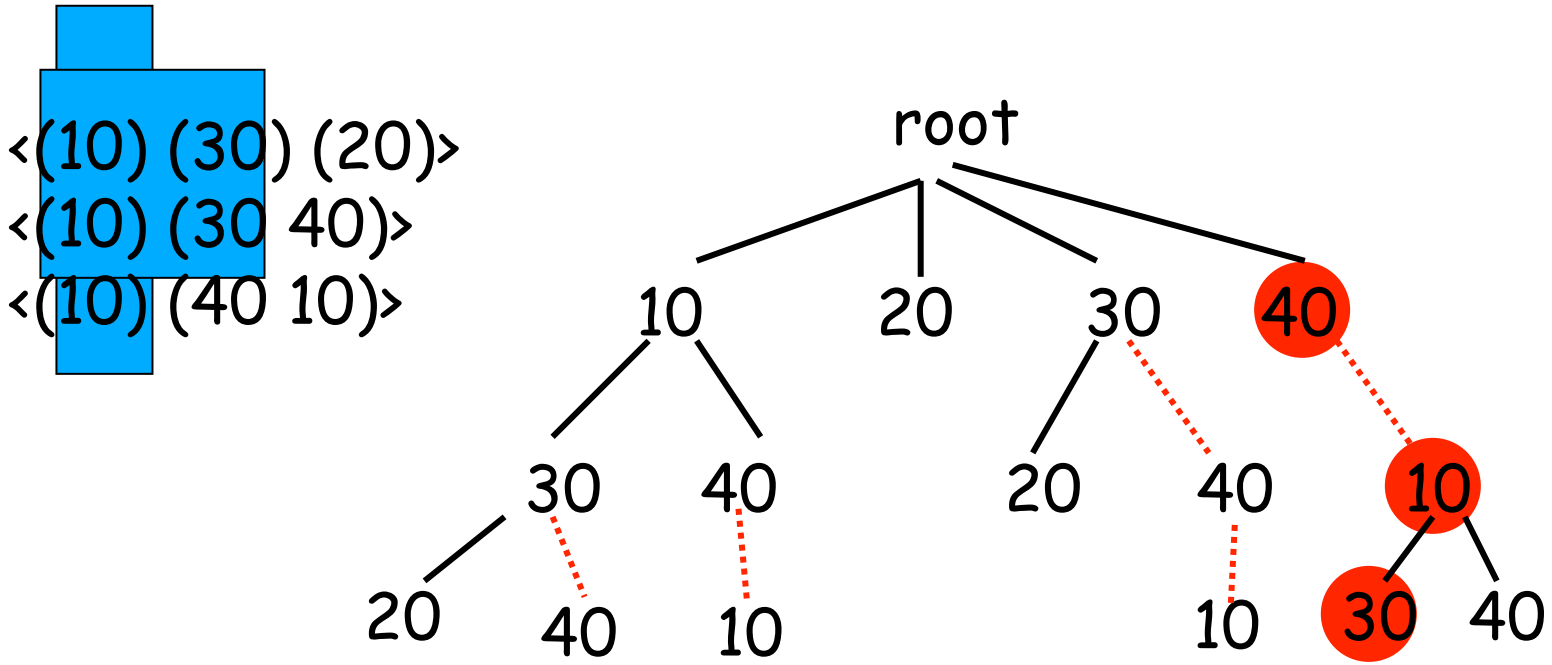
# Stockage des candidats




S = <(10) (30) (10 40)>

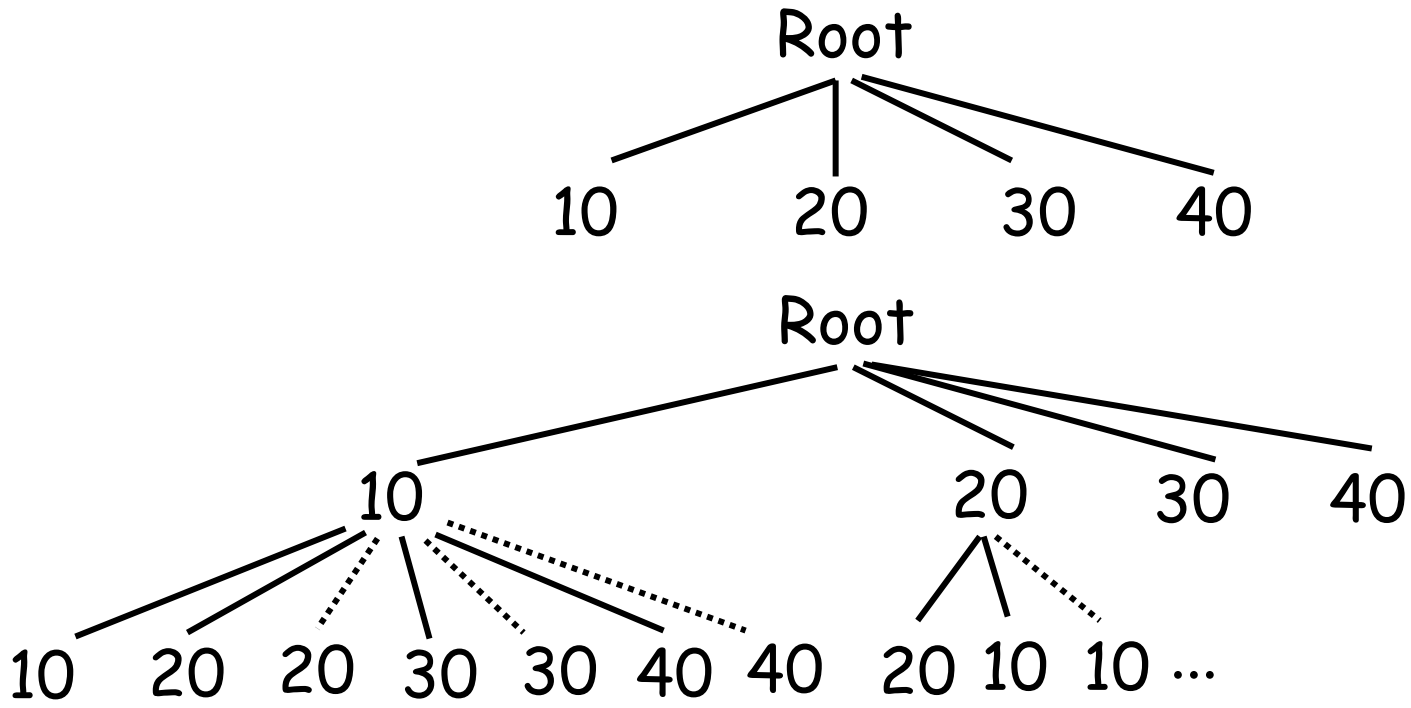
# PSP (Prefix Tree for Sequential Patterns)

- Vers une structure plus efficace : prefix tree



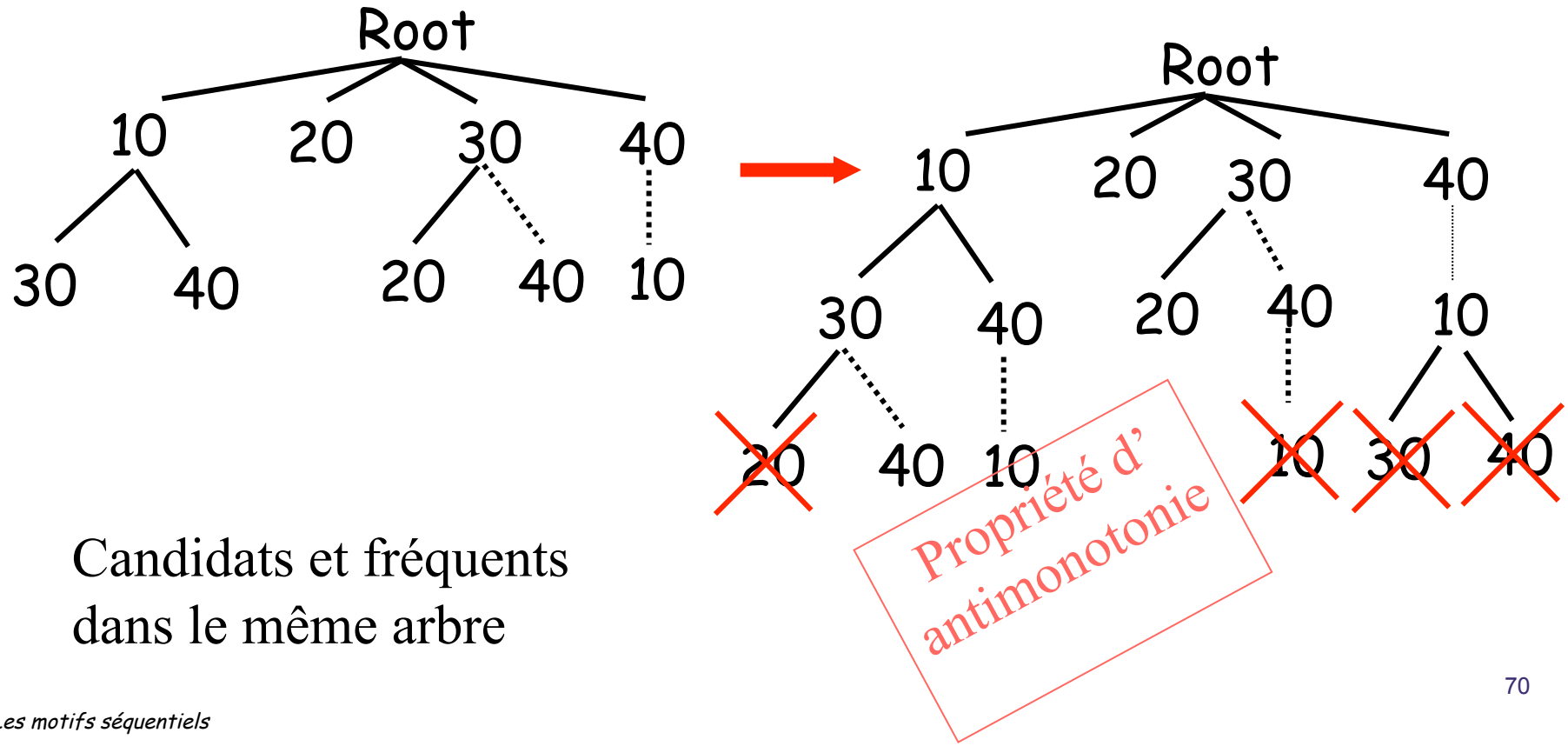
# PSP (cont.)

- Génération des candidats de taille 2



# PSP (cont.)

- Génération des candidats de taille > 2




Candidats et fréquents
dans le même arbre

# SPAM

- Utilisation de bitmaps pour rechercher les motifs fréquents
- Hypothèse : la base tient toujours en mémoire
- On construit d'un arbre lexicographique contenant toutes les branches possibles – élimination des branches en fonction du support
- Nouvelle représentation des données

# SPAM (cont.)

- Représentation verticale des données

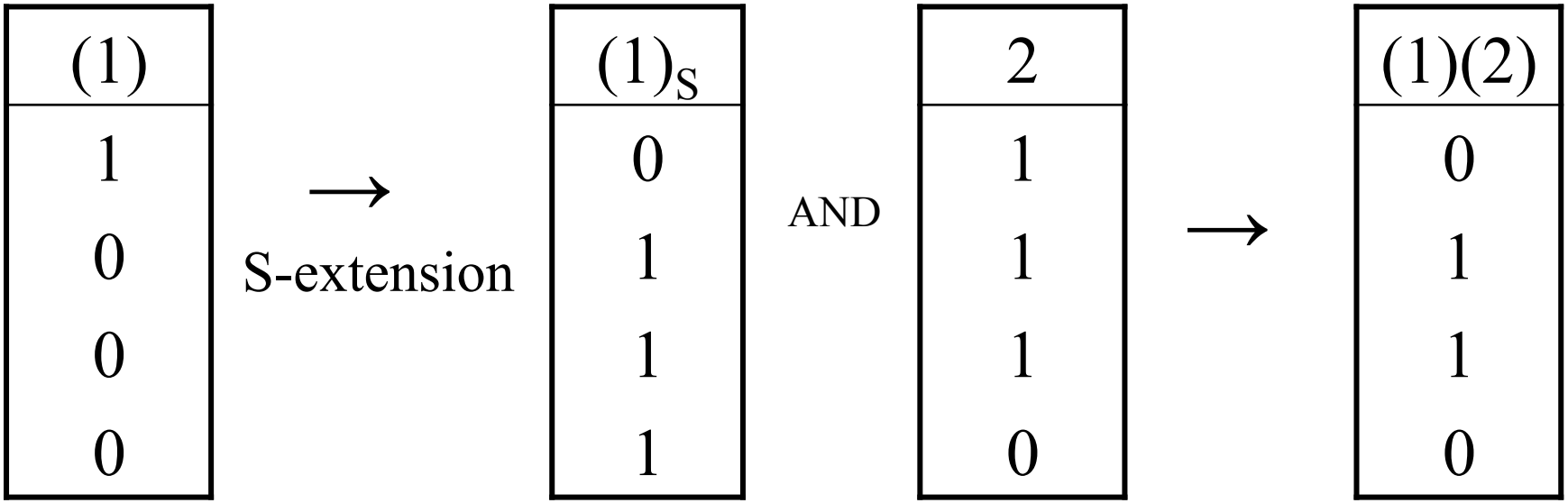
$C1 = <(1)_3\ (1)_5>$

| | | (1) |
|---|---|---|
| **C1** | **T1** | 0 |
| | **T2** | 0 |
| | **T3** | 1 |
| | **T4** | 0 |
| | **T5** | 1 |

- S-Extension
- I-Extension

# SPAM (cont.)

- S-Extension : un bitmap transformé + AND
- I-Extension : AND
- Exemple : recherche du candidat (1) (2)



# Conclusions

- Depuis 1996 :
- Problème de recherche ouvert
- Données de plus en plus complexes (représentations, …), obtenues de plus en plus rapidement (incrémental, flots de données), avec de nouvelles contraintes (préservation de la vie privée, contraintes de dimensions, temporelles), avec valeurs manquantes, …
- Besoins de nouveaux indicateurs de qualité

# Conclusions

- Une URL : KDD Mine ttp://www.kdnuggets.com
- Google, citeseer, …
- Quelques outils

  Intelligent Miner (www.ibm.com)

  Entreprise Miner (SAS Institute)

  MineSet (Silicon Graphics Inc.)

  Clementine (Integral Solutions Ltd, racheté par SPSS)

  DBMiner (www.dbminer.com)
- Le projet Weka (librairie de classes Java)

  http://www.cs.waikato.ac.nz/ml/weka

# Références


- R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. SIGMOD'93, 207-216, Washington, D.C.
- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94 487-499, Santiago, Chile.
- R. Agrawal and R. Srikant "Mining sequential patterns", In Proc. ICDE'95, Taiwan, March 1995.
- R.J. Bayardo. Efficiently mining long patterns from databases. In *Proc. SIGMOD'98*, WA, June 1998
- S. Brin R. Motwani, J. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD'97*
- M.N. Garofalakis, R. Rastogi, K. Shim: SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. VLDB 1999: 223-234, Edinburgh, Scotland.
- J. Han, J. Pei, and Y. Yin: "Mining frequent patterns without candidate generation". In Proc. ACM-SIGMOD'2000, pp. 1-12, Dallas, TX, May 2000.
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'94, 181-192, Seattle, WA, July 1994.
- J. Han and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2000.
- J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.-C. Hsu, "FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining", In Proc. 2000 Int. Conf. on Knowledge Discovery and Data Mining (KDD'00), Boston, MA, August 2000.
- J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.-C. Hsu, "FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining", In Proc. 2000 Int. Conf. on Knowledge Discovery and Data Mining (KDD'00), Boston, MA, August 2000.
- J. Han, J. Pei, and Y. Yin: "Mining frequent patterns without candidate generation". In ACM-SIGMOD'2000, Dallas, TX, May 2000.
- V. Kapoor, P. Poncelet, F. Trousset and M. Teisseire. "Privacy Preserving Sequential Pattern Mining in Distributed Databases". Proceedings of the Fifteenth Conference on Information and Knowledge Management (CIKM 2006), Arlington, US, November 2006.

# Références


- H. Mannila, H. Toivonen and A.I. Verkamo. Efficient algorithms for discovering association rules. In *Proc. KDD'94*, WA, July 1994
- P.A. Laur, M. Teisseire and P. Poncelet. "AUSMS: An Environment for Frequent Sub-Substructures Extraction in a Semi-Structured Object Collection". Proceedings of the 14th International Conference on Database and Expert Systems Applications (DEXA'03), Prague, Czech Republic, LNCS, pages 38-45, September 03.
- F. Masseglia, P. Poncelet and M. Teisseire. "Peer-to-Peer Usage Mining: a Distributed Mining Approach". Proceedings of the IEEE 20th International Conference on Advanced Information Networking and Applications (AINA 2006), Vienna, Austria, April 2006.
- F. Masseglia, F. Cathala and P. Poncelet. "PSP: Prefix Tree For Sequential Patterns". Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98), Nantes, France, LNAI, Vol. 1510, pp. 176-184, September 1998.
- F. Masseglia, M. Teisseire et P. Poncelet. "Extraction de motifs séquentiels - Problèmes et Méthodes". Revue Ingénierie des Systèmes d'Information (ISI), Numéro spécial "Extraction et usages multiples de motifs dans les Bases de Données", Vol.9, N. 3-4, 2004, pp.183-210.
- F. Masseglia, M. Teisseire and P. Poncelet. "Pre-Processing Time Constraints for Efficiently Mining Generalized Sequential Patterns". Proceedings of the 11th International Symposium on Temporal Representation and Reasoning (TIME'04), Tatihou, Basse Normandie, France, July 2004
- F. Masseglia, P. Poncelet and M. Teisseire. "Incremental Mining of Sequential Patterns in Large Databases". Data and Knowledge Engineering, Volume 46, Issue 1, pages 97-121, 2003.(PDF)
- F. Masseglia, M. Teisseire and P. Poncelet. "HDM: A Client/Server/Engine Architecture for Real Time Web Usage". Knowledge and Information Systems (KAIS) journal, Vol. 5, N° 4, October 2003.

# Références


- C. Raissi and P. Poncelet. "Towards a New Approach for Mining Maximal Frequent Itemsets over Data Stream". Journal of Intelligent Information Systems, Springer (to appear 2006)
- C. Raissi, P. Poncelet and M. Teisseire. "SPEED: Mining Maximal Sequential Patterns over Data Streams". Proceedings of the 3rd IEEE International Conference on Intelligent Systems (IEEE IS 2006), London, UK, September 2006.
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association in large databases. In *VLDB'95*
- R. Srikant and R. Agrawal "Mining sequential patterns: Generations and performance improvements", In Proc. EDBT'96, France, March 1996.
- H. Toivonen. Sampling large databases for association rules. In *VLDB'96*
- Wei Wang, Jiong Yang, Philip S. Yu: Mining Patterns in Long Sequential Data with Noise. SIGKDD Explorations 2(2): 28-33 (2000)
- M.J. Zaki. Efficient enumeration of frequent sequences. CIKM'98. Novermber 1998.