La représentation des images par les graphes

Aline Deruyver

Laboratoire Icube, équipe CSTB

Plan du cours

- Pourquoi les graphes ?
- Rappels et définitions
- De l'image à l 'objet graphe: les graphes d'adjacence de régions
- Simplification de graphes
  - division et fusion de régions
  - fusion pyramidale
- Interprétation d 'images: Mise en correspondance de graphes
  - Isomorphisme de graphes et sous graphes
  - Isomorphisme de sous-graphe avec tolérance d 'erreur
  - Mise en correspondance inexacte: un problème de satisfaction de contraintes

Pourquoi des graphes ?

- Intérêt : représentation compacte, structurée, complète, facile à manipuler
- Applications :
  - Traitement d'images : segmentation, détection de contours
  - Reconnaissance des formes : caractères, objets (bâtiments 2D ou 3D,
  - structures cérébrales, reconnaissance de visages (avec ou sans modèle)
  - Recalage d'images
  - Indexation
  - Interprétation de scènes structurées **…**

Définition et rappels

- Graphe : G = (X,E)
  - X ensemble des sommets (|X| ordre du graphe)
  - E ensemble des arêtes (|E| taille du graphe)
  - graphe complet (taille n(n−1)/2)
  - graphe partiel G = (X,E ') avec E ' partie de E
  - sous-graphe F = (Y,E '), $Y \subseteq X$ et $E' \subseteq E$
  - degré d'un sommet x : d(x) = nombre d'arêtes
  - graphe connexe : pour toute paire de sommets, il existe une chaîne les reliant
  - arbre : graphe connexe sans cycles
  - clique : sous-graphe complet
  - graphe dual (face → noeud)
  - graphe aux arêtes (arête → noeud)
  - hypergraphe (relations n-aires)
  - graphes pondérés : coûts ou poids sur les arcs

Exemples de graphes

- Graphe relationnel attribué : $G = (X, E, \mu, \nu)$
  - $\mu : X \rightarrow LX$ interpréteur de sommets (LX = attributs des noeuds)

    $\nu : E \rightarrow LE$ interpréteur d'arcs (LE = attributs des arcs)
  - Exemples :
    - graphe des pixels
    - graphe d'adjacence de régions
    - régions de Voronoï / triangulation de Delaunay
    - graphe de primitives avec des relations plus complexes
- Graphe aléatoire : sommets et arcs = variables aléatoires
- Graphe flou : $G = (X, E = X \times X, \mu f, \nu f)$
  - $\mu f : X \rightarrow [0, 1]$

    $\nu f : E \rightarrow [0, 1]$
  - avec $\forall (u, v) \in X \times X \quad \nu f(u, v) \leq \mu f(u) \mu f(v)$ ou $\nu f(u, v) \leq \min[\mu f(u) \mu f(v)]$

Exemples de graphes (suite)

- Graphe d'attributs flous : graphe relationnel attribué avec valeur floue pour chaque attribut
- Graphe hiérarchique :
  - graphes à plusieurs niveaux et graphe biparti entre deux niveaux
  - (approches multi-échelles, regroupements d'objets, ...)
  - Exemples :
    - quadtrees, octrees
    - pyramide de graphes d 'adjacence de régions
- Graphes de raisonnement
  - (graphe de mise en correspondance, arbre de décision,...)

Plan du cours

- Pourquoi les graphes ?
- Rappels et définitions
- De l'image à l 'objet graphe: les graphes d'adjacence de régions
- Simplification de graphes
  - division et fusion de régions
  - fusion pyramidale
- Interprétation d 'images: Mise en correspondance de graphes
  - Isomorphisme de graphes et sous graphes
  - Isomorphisme de sous-graphe avec tolérance d 'erreur
  - Mise en correspondance inexacte: un problème de satisfaction de contraintes

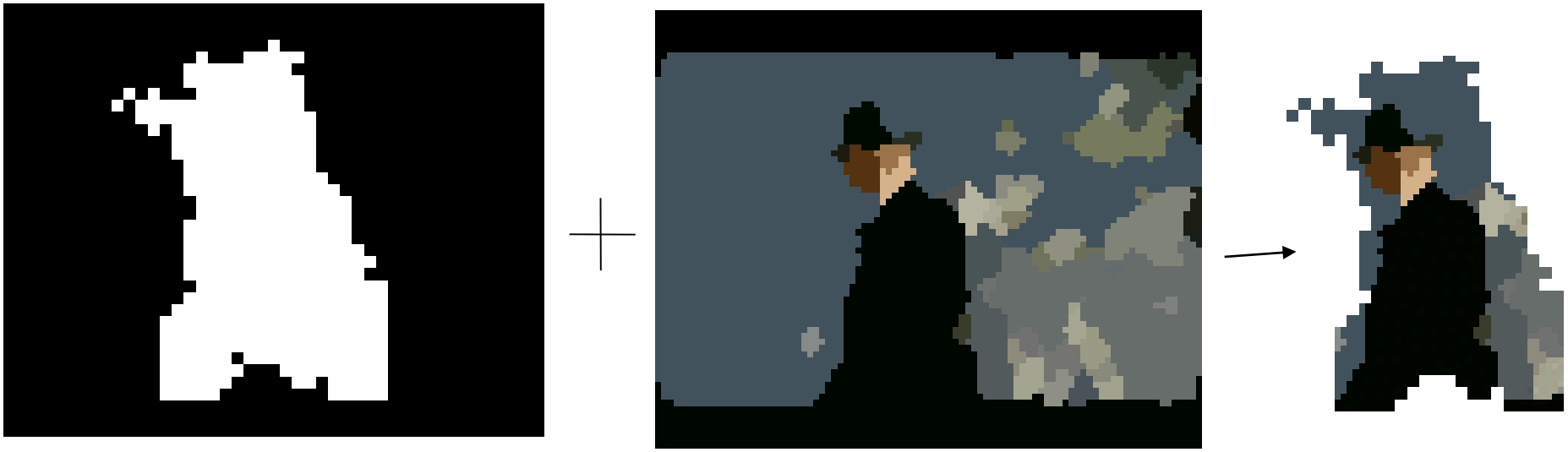
De l'image à l'objet-graphe

- **Segmentation**

Manerbba, Benois-Pineau, Leonardi : *Extraction of foreground objects from a mpeg2 video stream in rough indexing framework*, SPIE '04, San José, California

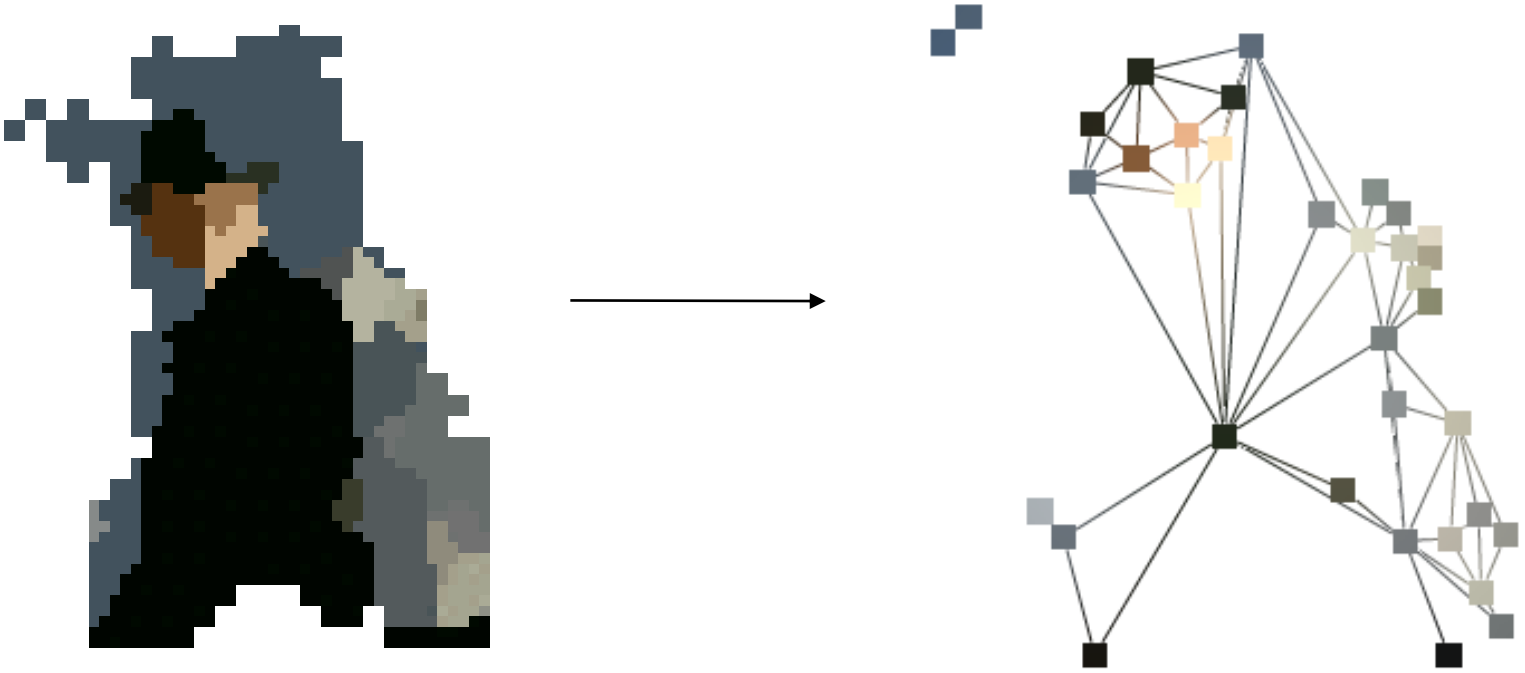
De l’image à l’objet-graphe

- Segmentation
- **Récupération et application du masque de l'objet**



De l’image à l’objet-graphe

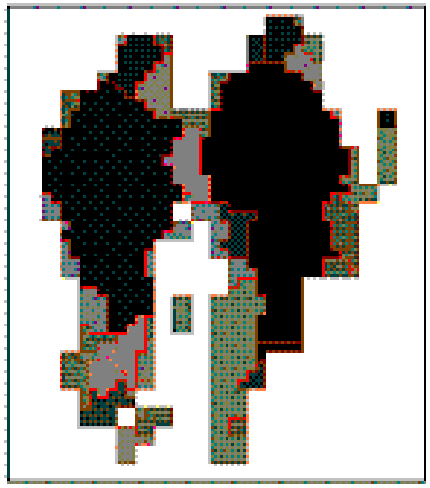
- Segmentation
- Récupération et application du masque de l'objet
- **Construction du graphe d'adjacence**

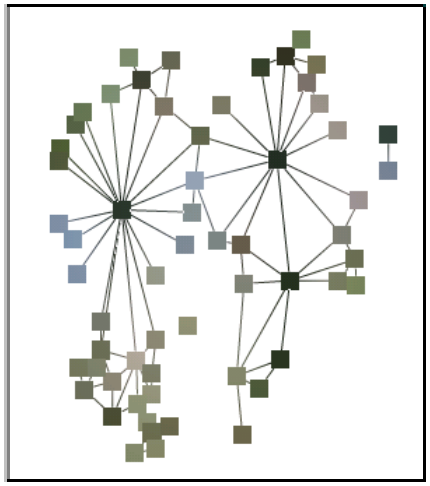


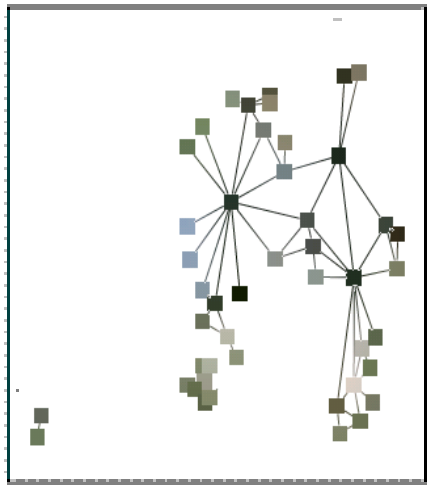
De l'image à l'objet-graphe

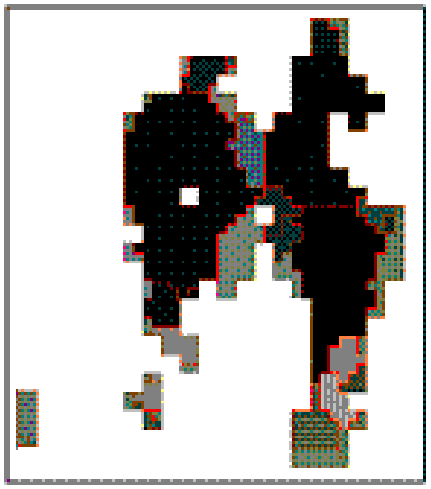
60 régions

46 régions









De l'image à l'objet-graphe

- Défauts

  – Sursegmentation

    ⇨ beaucoup de petites régions

  – Problèmes liés à l'extraction du masque

    ⇨ pixels du fond

    ⇨ régions coupées

Plan du cours

- Pourquoi les graphes ?
- Rappels et définitions
- De l'image à l 'objet graphe: les graphes d'adjacence de régions
- Simplification de graphes
  - division et fusion de régions
  - fusion pyramidale
- Interprétation d 'images: Mise en correspondance de graphes
  - Isomorphisme de graphes et sous graphes
  - Isomorphisme de sous-graphe avec tolérance d 'erreur
  - Mise en correspondance inexacte: un problème de satisfaction de contraintes

Division et fusion de régions dans un graphe

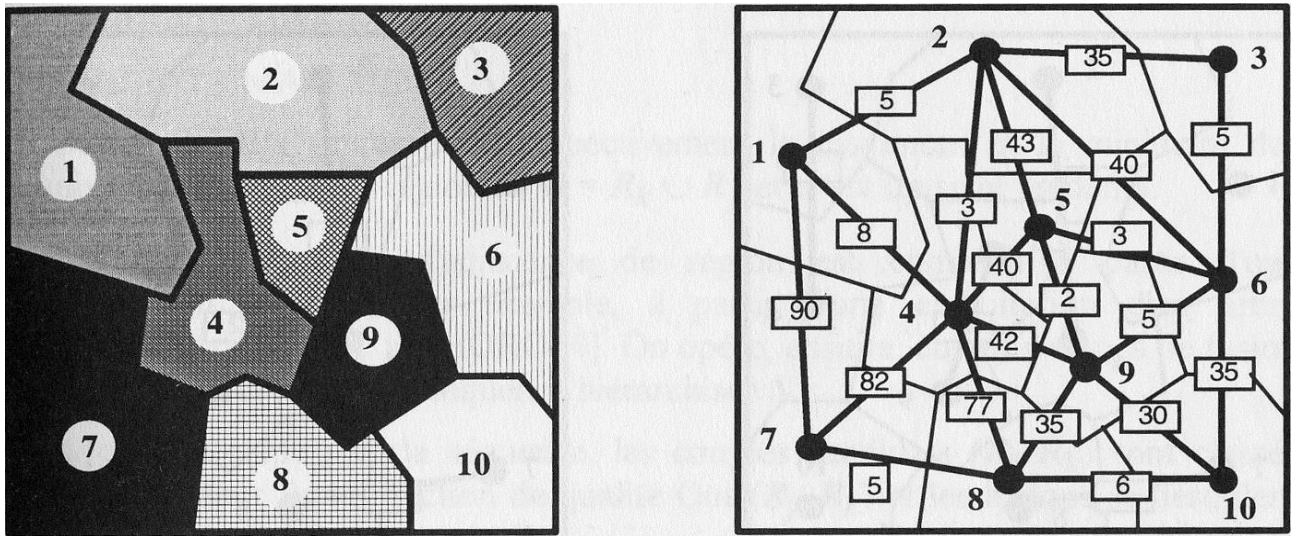
- Procède par éclatement et regroupement de composantes de l'image décrite à l'aide d'un graphe d'adjacence.

- Définition: Un graphe d'adjacence de régions est défini par G=(V,E)
  - V: ensemble des sommets
  - E: ensemble des arêtes

  Chaque sommet vi est associé à une région Ri. Une arête est un couple (vj,vk) de sommets entre 2 régions adjacentes. G est non orienté.

Division et fusion de régions

- Soit une segmentation initiale est représentée par un tel graphe



Division et fusion de régions: algorithme général (Pavlidis 77)

- Module division
  - T := VRAI
  - Tant que t répéter
    - t:= faux
    - pour chaque sommet Rk faire
      - si non(Pred1(Rk)) alors
        - » t:=VRAI
        - » partager Rk en Nk régions suivant un critère donné
        - » mettre à jour le graphe
      - Fsi
    - Fpour
  - ftq

Division et fusion de régions: algorithme général (Pavlidis 77)

- Module de fusion
  - T:= VRAI
  - Tant que t répéter
    - T:= FAUX
    - Pour chaque sommet Rk adjacent à un sommet Rj faire
      - Si Pred2(Rk $\cup$ Rj)=VRAI alors
        - » T:=vrai
        - » Fusionner Rj et Rk
        - » Mettre à jour le graphe
      - Fsi
    - Fpour
  - ftq

Division et fusion de régions: convergence

- La procédure de division converge: le graphe limite est celui correspondant au maillage
- La procédure de fusion converge: le graphe limite possède un seul sommet correspondant à toute l'image
- Pas de cycle possible: toutes les fusions ont lieu avant les divisions

Division et fusion de régions: charge de calcul

- Elle est liée:
  - Au calcul des attributs associés aux sommets et aux arêtes du graphe
  - À la mise à jour du graphe après chaque fusion ou division. Cette mise à jour est généralement moins coûteuse après une fusion.
  - Exemple: si pour un attribut A(Ri) il existe une loi de composition telle que

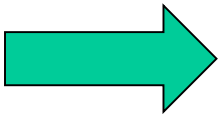
$$A(Ri \cup Rj) = A(Ri) \bigcirc A(Rj)$$

Division et fusion de régions: avantage de l'approche

- L'information obtenue à partir des régions est statistiquement plus fiable que celle calculée à partir des pixels (robustesse vis-à-vis du bruit).

Division et fusion de régions: problème de l'ordre des fusions

- Des ordres de fusion différents conduisent à des segmentations différentes.
- Solution: établir une hiérarchie de critères
- Exemple: associer à chaque arête de G[V,E]un coût de fusion.



Détecter sur le graphe l'arête de moindre coût à chaque itération. La segmentation est alors unique

Division et fusion de régions:sélection et fusion des arêtes candidates

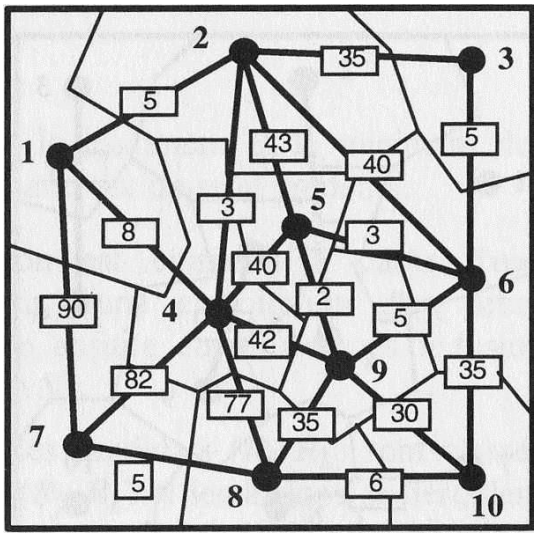
- Répéter
  - Sélectionner un sous-ensemble E’ de E tel que chaque sommet de V n’apparaisse qu’une fois dans E’
  - Fusionner 2 à 2 les régions associées aux éléments de E’
  - Mettre à jour V et E
- Jusqu’à plus de fusion possible

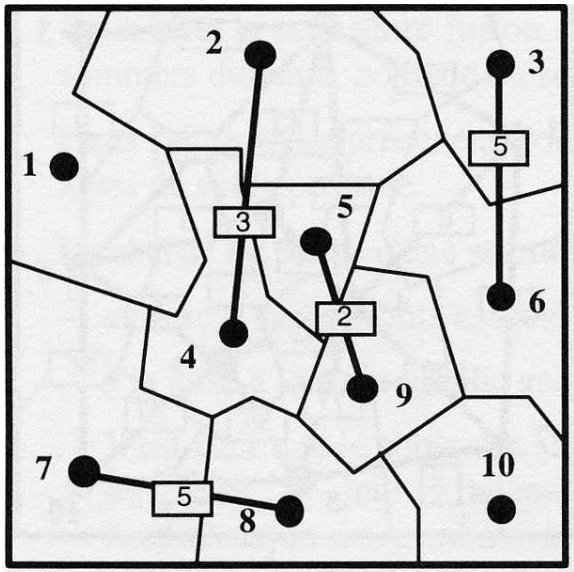
Division et fusion de régions: choix de E'

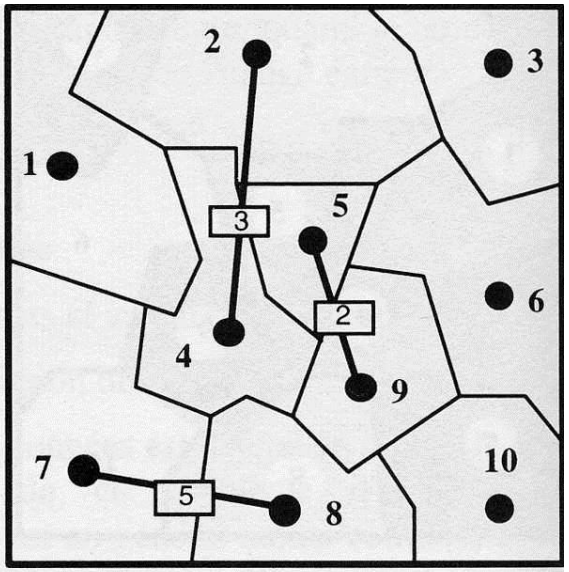
- But: garantir une bonne segmentation
- Exemples de choix:
  - Soit lc le coût minimale de fusion des arêtes et t un seuil donné. E' contient les arêtes de coût de fusion < lc+t
  - Test de reconnaissance mutuelle entre les régions: chaque sommet vj sélectionne l'arc vers le sommet adjacent vk de meilleur coût de fusion. Si vj est aussi le meilleur coût pour vk alors (vj,vk) $\in$ E'
  - Extraction itérative: Un seuil t sur le coût de fusion est choisi. Parmi les arêtes de coût <t on sélectionne l'arête (vj,vk) de moindre coût. On supprime les arêtes aboutissant aux sommets vj et vk. On sélectionne parmi les arêtes restantes celle de coût minimal et on recommence jusqu'à avoir traité toutes les arêtes.

Division et fusion de régions

Graphe d'origine







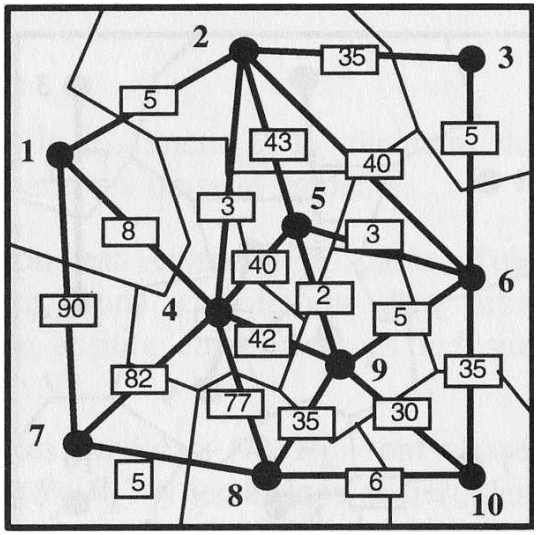
a) *minimums locaux stricts du coût de fusion* b) *accord mutuel entre sommets*

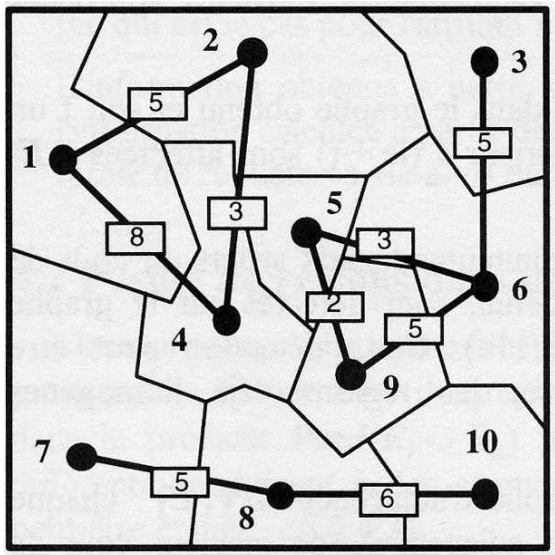
Figure 12.11 : Sélection de E', à partir de la figure 12.10 (seuil fusion = 5)

Résultat obtenu avec la technique de reconnaissance mutuelle. Seuil t=5

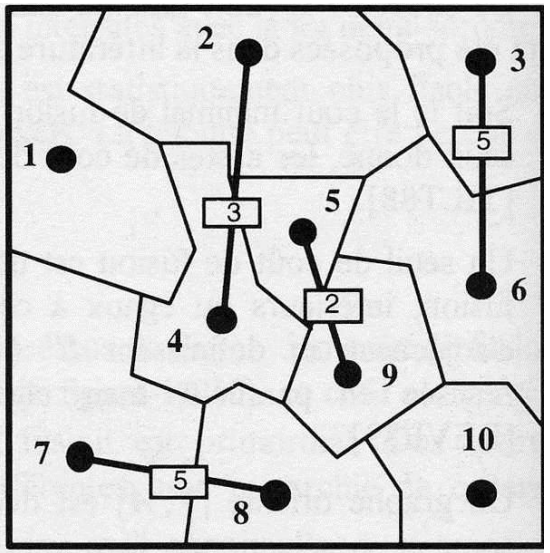
Division et fusion de régions

Graphe d'origine





a) *restriction de* E *aux arêtes de coût de fusion inférieur au seuil (=15)*



b) *résultat de l'extraction itérative des arêtes*

Résultat obtenu par extraction itérative des arêtes.
Seuil=15

Fusion multicritère

- Dans la phase de fusion un couple de région Rk,Rj devra:
  - Vérifier un prédicat d'uniformité ($Pred_i$)
  - Minimiser une fonction de qualité ($Qual_i$)
- Monga et Gagalowicz ont développé un algorithme qui utilise une séquence hiérarchisée de critères de fusion:
  - $S= \{(Pred_1, Qual_1), \ldots, (Pred_n, Qual_n)\}$

- On opère autant d'étapes de fusion qu'il y a de critères dans la séquence.

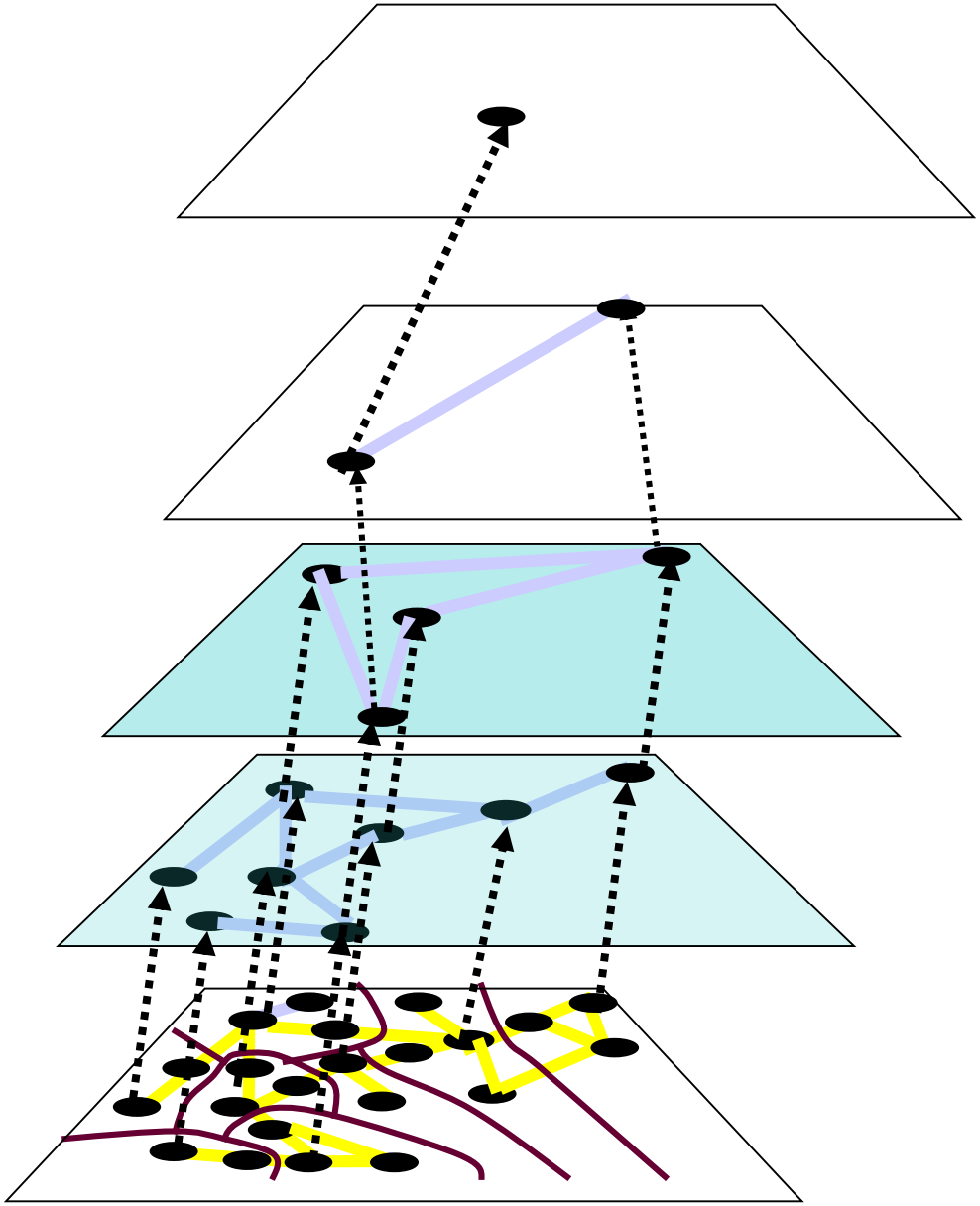
- A chaque étape de la séquence, les couples (Rk,Rj) sont classés suivant la valeur de la fonction de qualité et les fusions se déroulent suivant l'ordre établi.

Pyramides adaptatives

- Principe: partitionnement de l'image par un procédé de type fusion ascendante de régions.

- Initialisation:
  - Un pixel est un nœud du graphe
  - Les arêtes sont les liens de voisinage entre régions

- Processus itératif: à chaque itération on tente de supprimer le maximum de régions par agglomération (réduction de graphe)



Fusion pyramidale

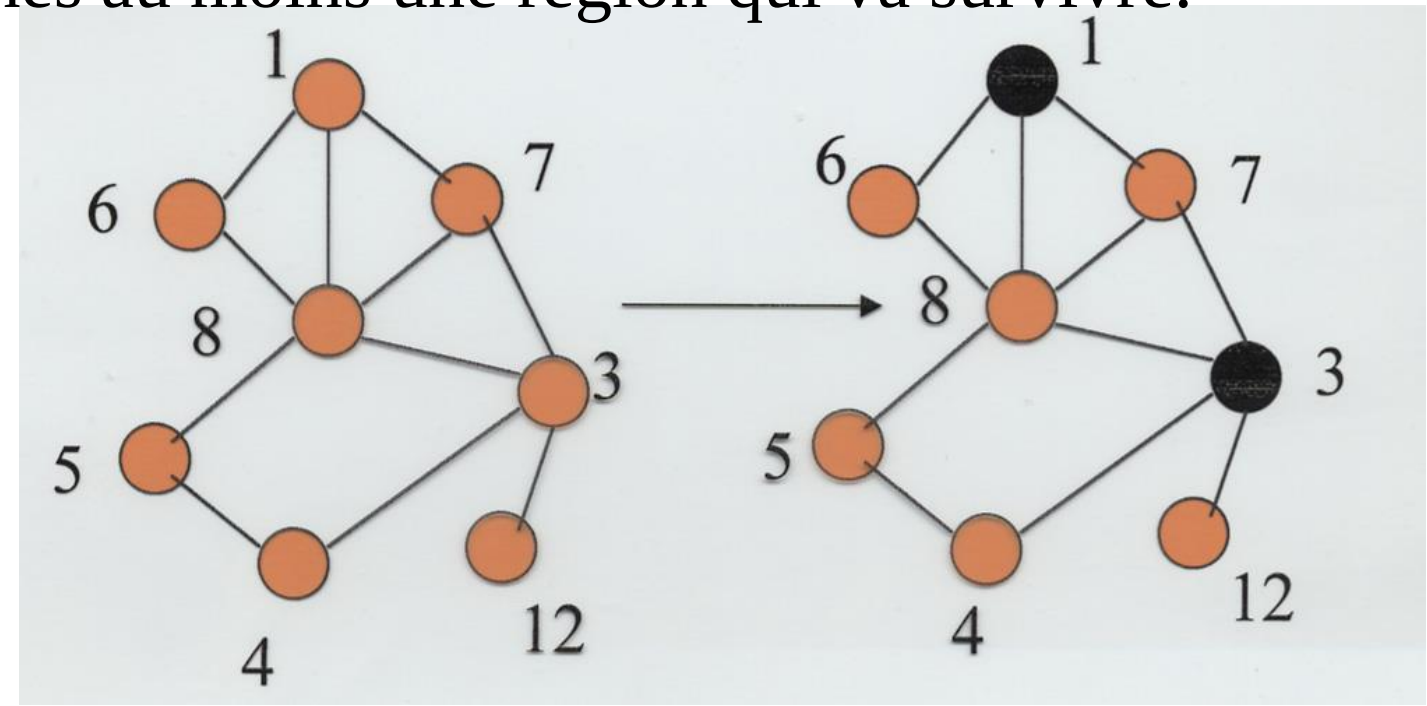
Pyramides adaptatives

- Règles de fusion:
  - Sélectionner un sous-ensemble du graphe: régions survivantes
  - Affectation de toutes les régions non survivantes à une région survivante en conservant la cohérence du graphe
- Les valeurs associées aux nœuds pourront être une mesure de variation (variance). Les régions survivantes seront des minimums locaux (régions homogènes)
- Les régions non survivantes fusionneront avec la région survivante voisine qui leurs ressemble le plus

Pyramides adaptatives

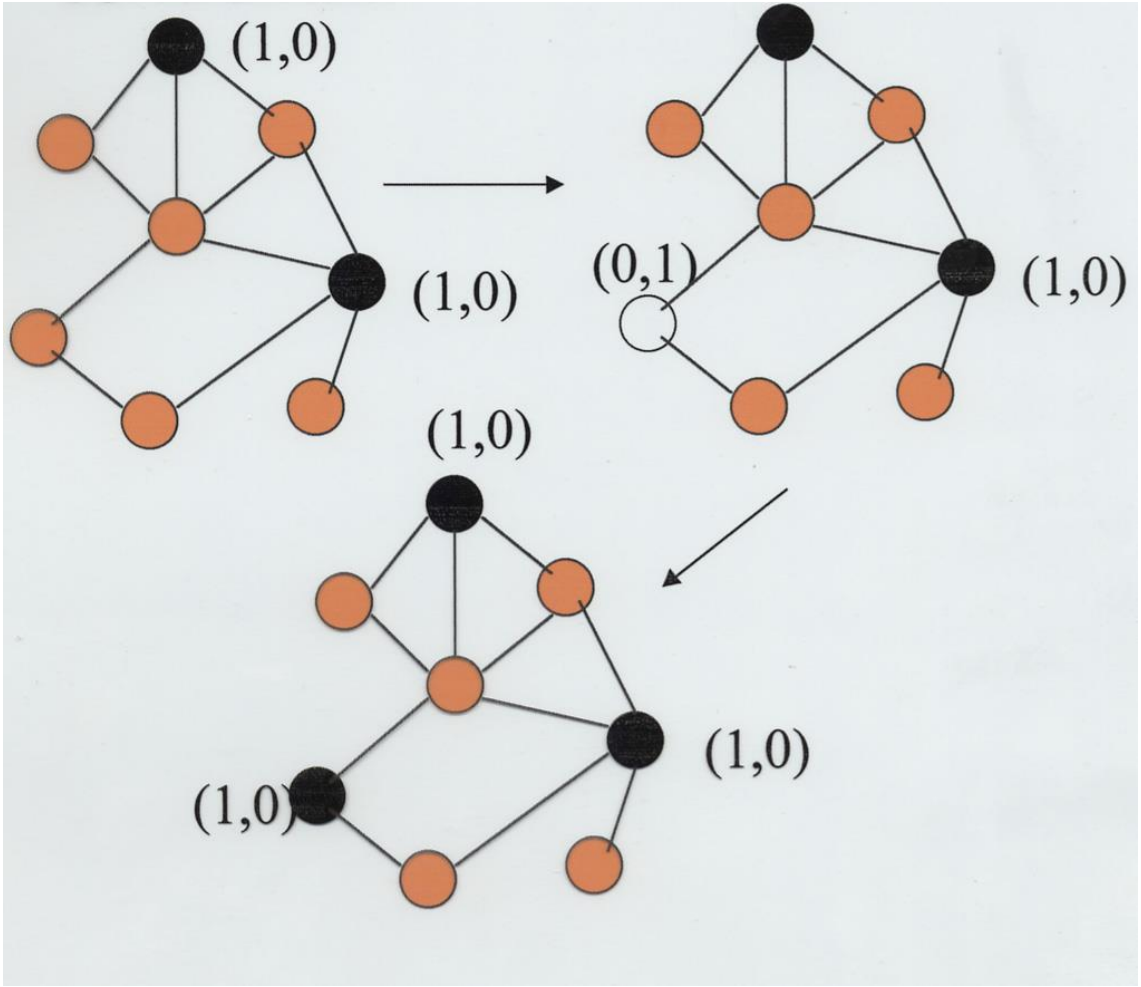
Contraintes:

- deux régions voisines ne peuvent survivre toutes les deux.
- toute région non survivante possède, parmi ses régions voisines au moins une région qui va survivre.



Pyramides adaptatives

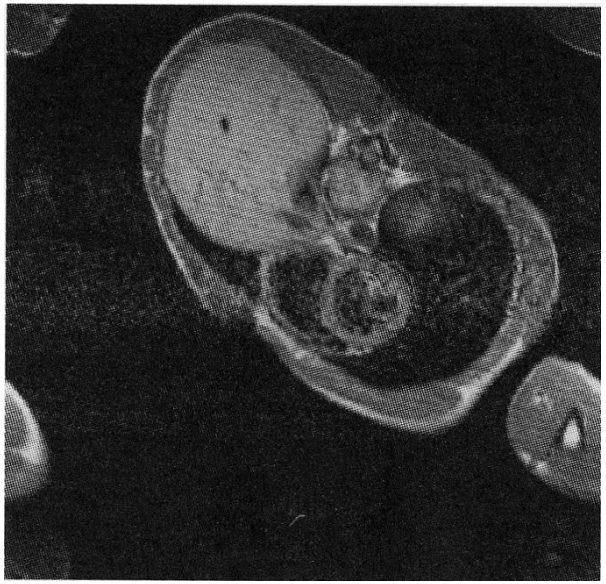
Etape de décimation: satisfaction de la deuxième contrainte:

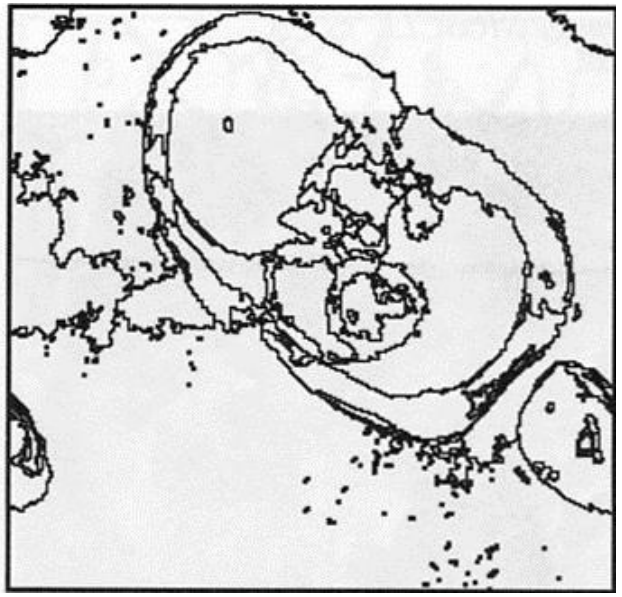


Pyramides adaptatives: neutralisation de certaines régions

- Si P est non survivante et si P’ est la plus similaire des ses régions voisines survivantes alors
  - Si contraste(P,P’) > seuil alors
    - P est neutralisée
  - Sinon P fusionne avec P’

Pyramide adaptatives





Plan du cours

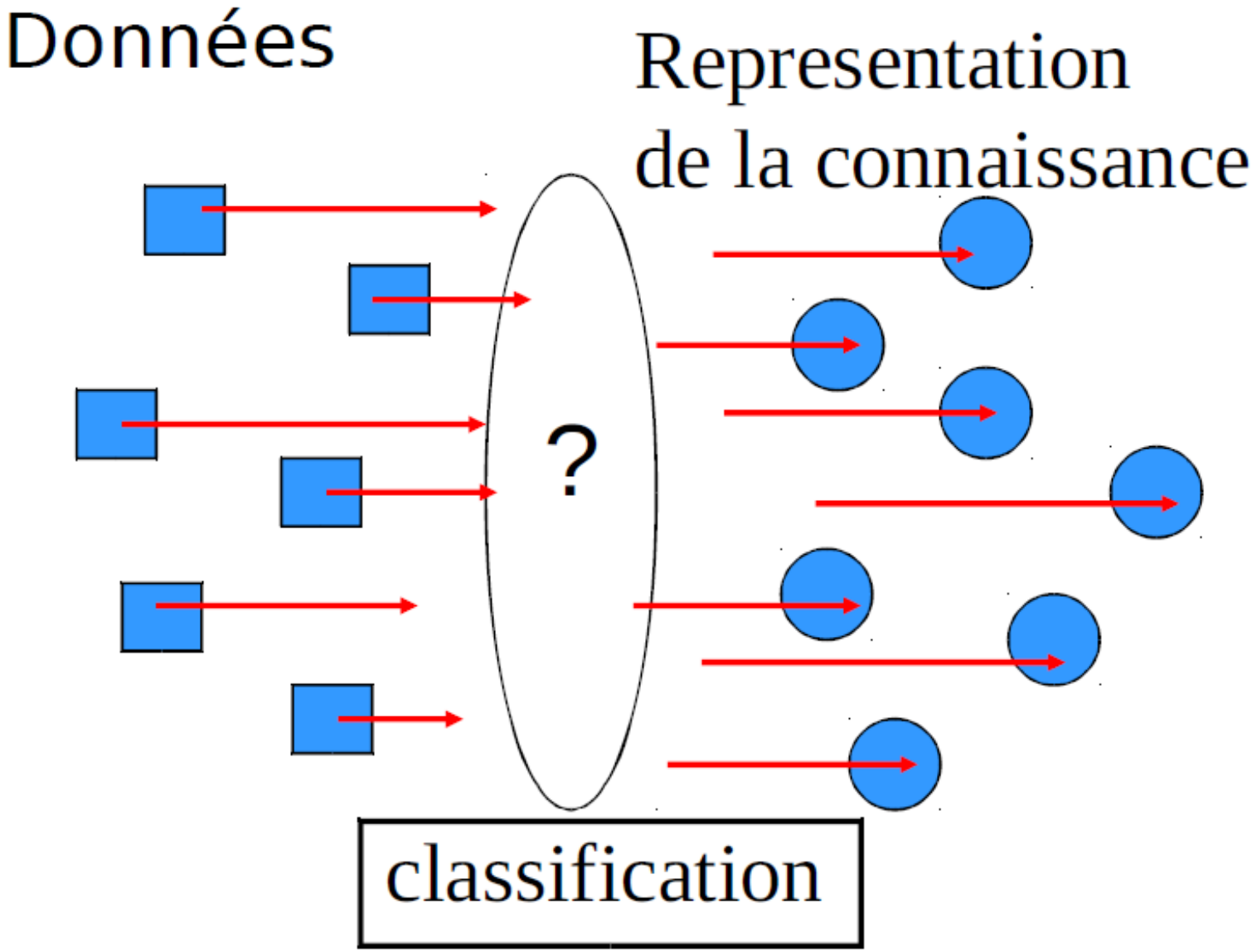
- Pourquoi les graphes ?
- Rappels et définitions
- De l'image à l 'objet graphe: les graphes d'adjacence de régions
- Simplification de graphes
  - division et fusion de régions
  - fusion pyramidale
- Interprétation d 'images: Mise en correspondance de graphes
  - Isomorphisme de graphes et sous graphes
  - Isomorphisme de sous-graphe avec tolérance d 'erreur
  - Mise en correspondance inexacte: un problème de satisfaction de contraintes

Reconnaissance des formes

- Objet : défini par un ensemble de primitives (noeuds du graphe)
- Relations binaires de compatibilité entre primitives (arcs du graphe)
- Clique : sous-ensemble de primitives compatibles 2 à 2 = configuration possible de l'objet
- Reconnaissance par détection de la clique maximale
- Recherche des cliques :
  - Problème NP-complet
  - Pas d'algorithme qui soit dans tous les cas non exponentiel
  - Construction d'un arbre de décision : un noeud de l'arbre = 1 clique du graphe
  - Elagage de l'arbre pour ne pas réengendrer les mêmes cliques

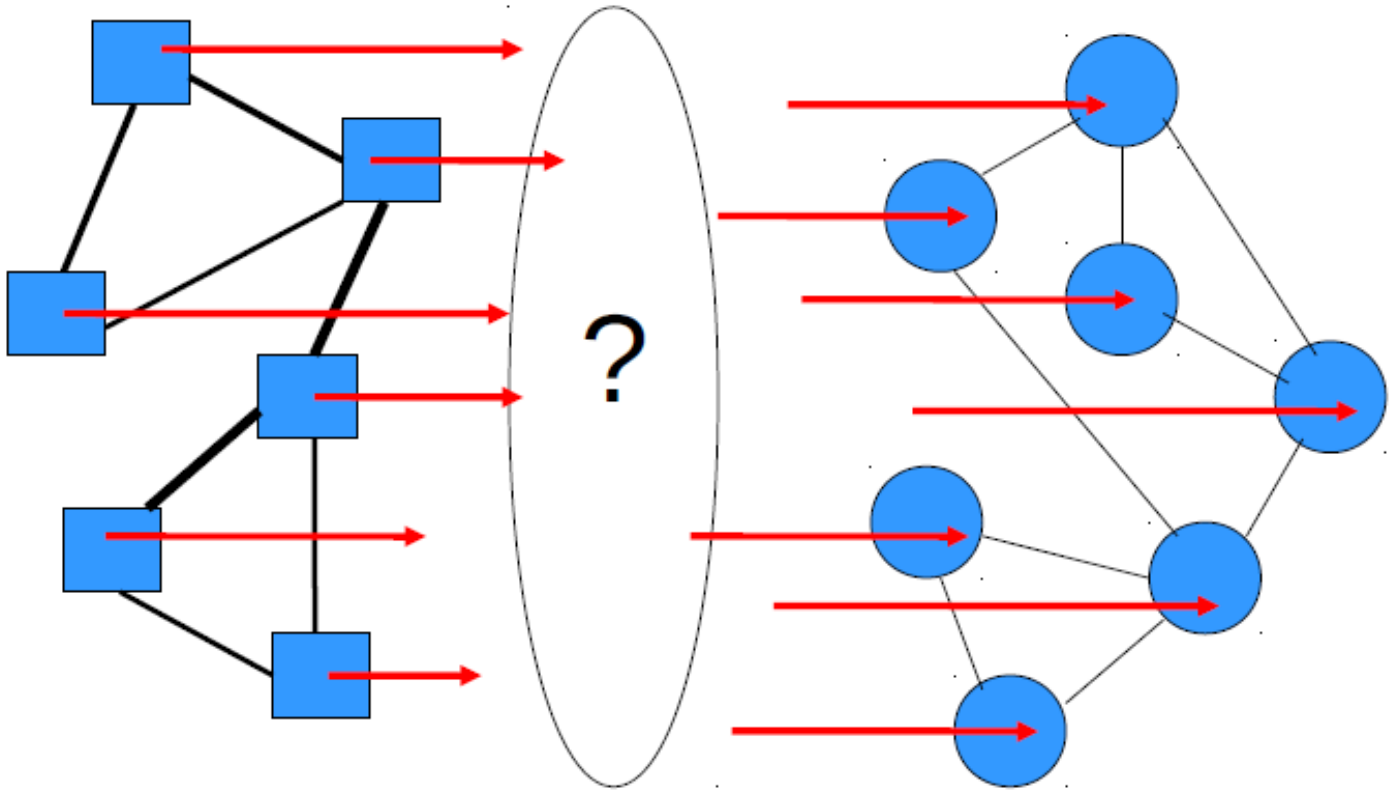
Interprétation d'images

- Tenenbaum et Barrow (1977)
  - Segmentation en régions
  - Construction du graphe d'adjacence des régions
  - Interprétation contrainte par un ensemble de règles sur :
    - 1. les objets (taille, couleur, texture,...)
    - 2. les relations entre objets (au-dessus de, à l'intérieur de, à côté de ...)


Données
Representation
de la connaissance
?
classification

Données

Representation de
La connaissance



Mise en correspondance

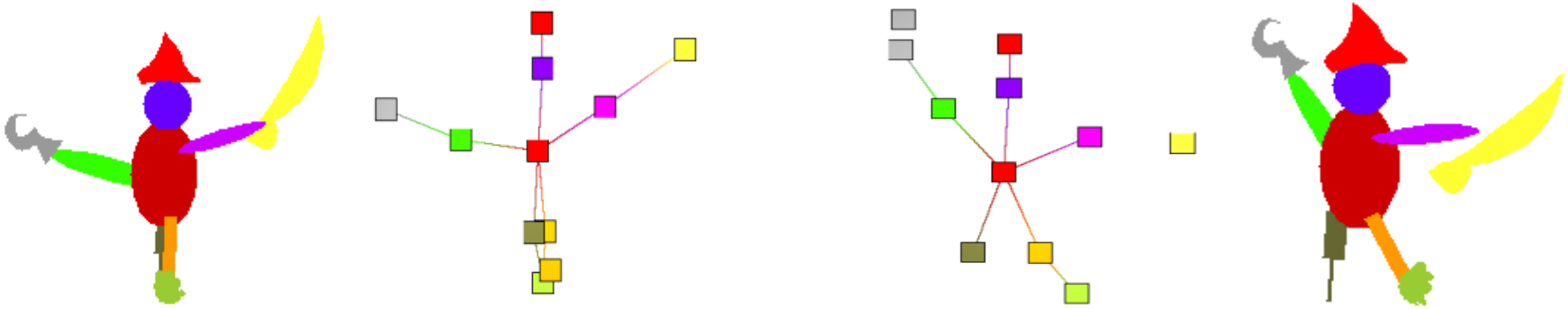
Mise en correspondance

- Soient deux graphes *G(V,E)* et *G'(V',E')*

mise en correspondance de P V avec **P'** **V'**

$$\|\boldsymbol{\nu}| - |\boldsymbol{\nu}'\| \leq \varepsilon$$

ε un entier positif



Plan du cours

- Pourquoi les graphes ?
- Rappels et définitions
- De l'image à l 'objet graphe: les graphes d'adjacence de régions
- Simplification de graphes
  - division et fusion de régions
  - fusion pyramidale
- Interprétation d 'images: Mise en correspondance de graphes
  - Isomorphisme de graphes et sous graphes
  - Isomorphisme de sous-graphe avec tolérance d 'erreur
  - Mise en correspondance inexacte: un problème de satisfaction de contraintes

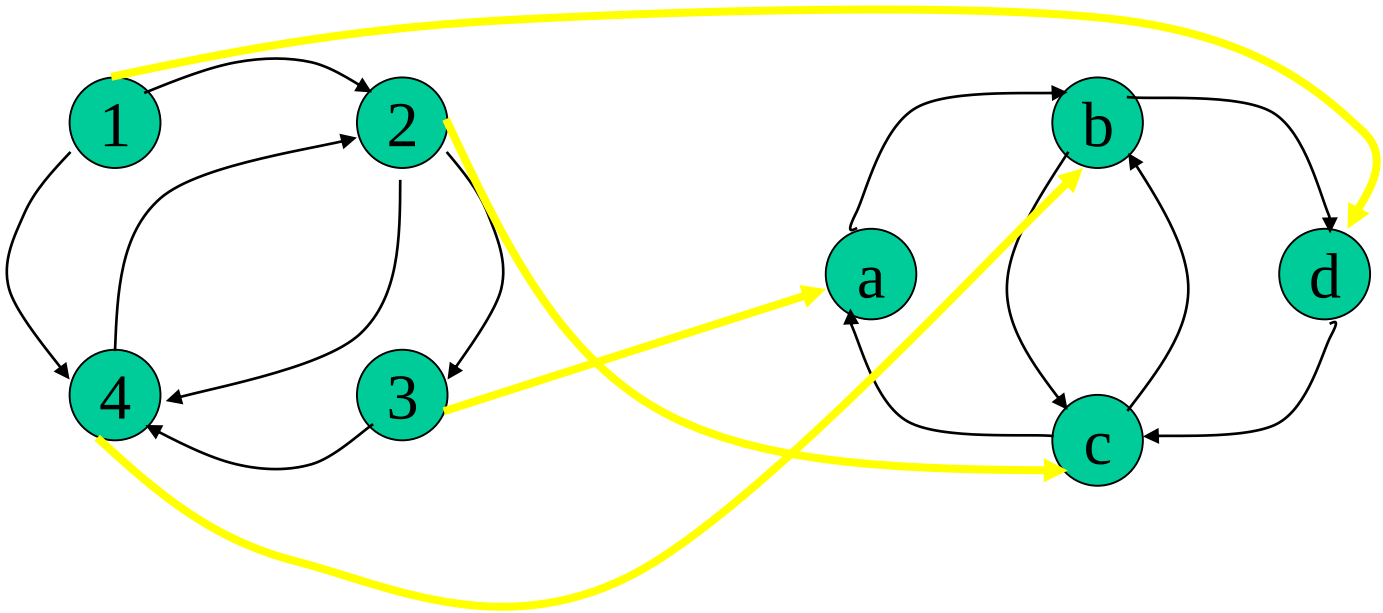
Mise en correspondance (matching) de graphes

- Problème :
  - Graphe(s) modèle(s) (atlas, carte, modèle(s) d'objet(s))
  - Graphe image construit à partir des données
  - Mise en correspondance des deux graphes
  - $G = (X, E, \mu, \nu) \rightarrow ? \; G' = (X', E', \mu', \nu')$
- Isomorphismes de graphes : fonction bijective $f : X \rightarrow X'$
  - $\mu(x) = \mu'(f(x))$

    $\forall e = (x1, x2), \exists e' = (f(x1), f(x2)) \; / \; \nu(e) = \nu'(e')$ et réciproquement

Isomorphisme de graphes

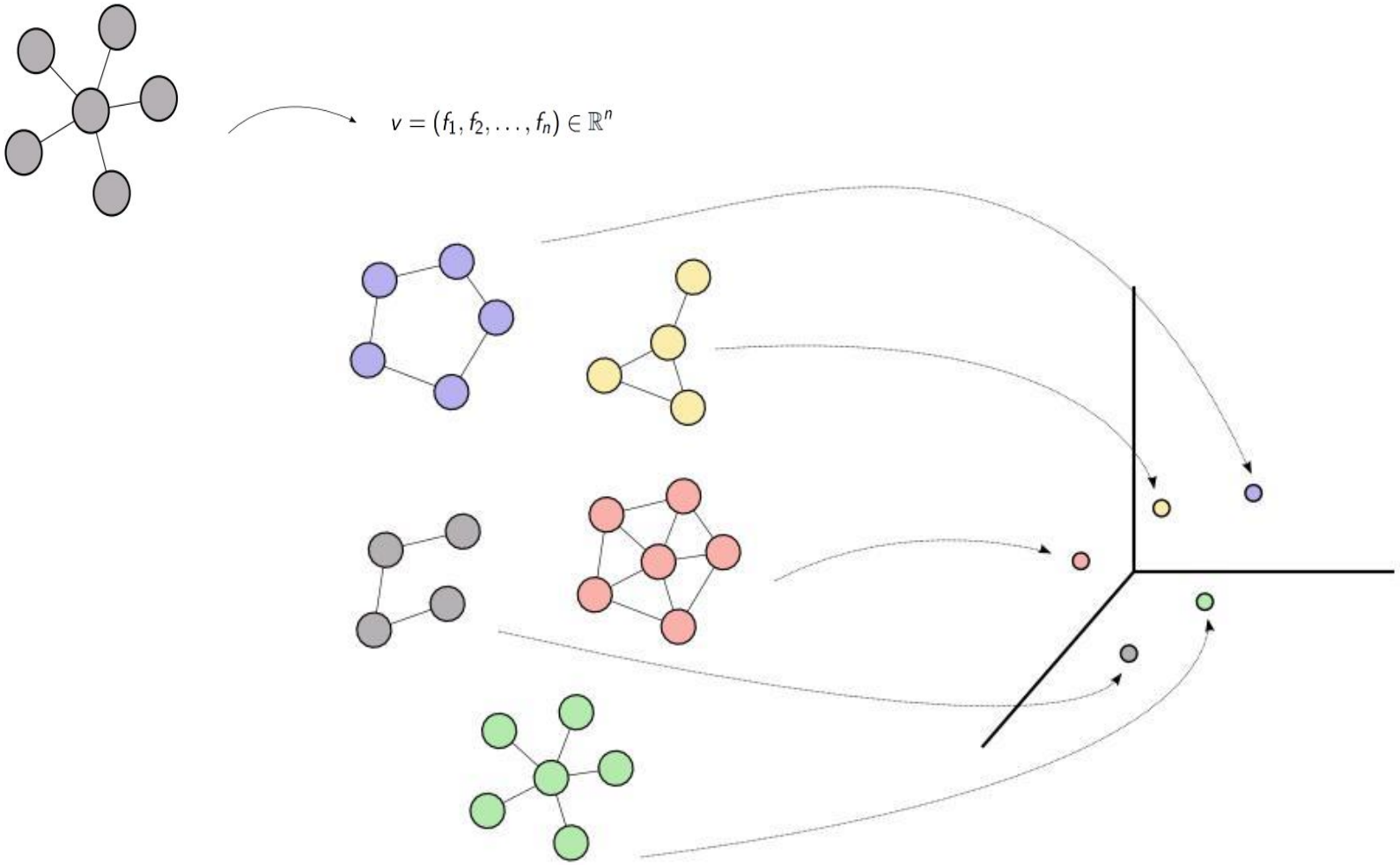
- Appariement de graphe, Exemple:



Appariement avec la fonction f

Souvent trop strict ⇒ isomorphismes de sous-graphes

Classification en terme de graphes



Classification en termes de graphes

- Soient X et Y les objets (resp. scènes), on cherche à savoir si

$$X \cong Y \text{ ou } X \subseteqq Y.$$

- L'isomorphisme de graphe $G = (V, E), G' = (V', E')$ $(X \cong Y)$
  - $|V| = |V'|$ et
  - il existe $\phi : V \to V'$ bijective tel que :

$$(v_1, v_2) \in E \Leftrightarrow (\phi(v_1), \phi(v_2)) \in E'$$

- L'isomorphisme partiel de sous graphes $(X \subseteqq Y)$
  - $|V| \leq |V'|$ et
  - il existe $\phi : V \to V'$ injective tel que :

$$(v_1, v_2) \in E \Rightarrow (\phi(v_1), \phi(v_2)) \in E'$$

Classification en termes de graphes

- L'isomorphisme de sous graphe
  - idem que isomorphisme partiel de sous graphe avec en plus :

$$\forall (v_1, v_2) \in V^2 \ (v_1, v_2) \notin E \Rightarrow (\phi(v_1), \phi(v_2)) \notin E'$$

on a donc :

$$(\phi(v_1), \phi(v_2)) \in E' \Rightarrow (v_1, v_2) \in E$$

Classification en termes de graphes

- Sous graphe partiel commun de taille maximum (mcps).

  graphe de taille maximum (en nombre de noeuds), sous graphe partiel de G et G'.

- Sous graphe commun de taille maximum (mcs)

  idem que mcps mais isomorphisme de sous graphe au lieu d'isomorphisme partiel de sous graphes.

Classification en termes de graphes

- Sous graphe (partiel) commun **maximum** ou **maximal** ?
- Étant donné un sous graphe (partiel) commun de G et G', celui-ci est dit **maximal**, si on ne peut plus y ajouter de sommet sans briser les isomorphismes.
- Un sous graphe est dit **maximum** si tout sous graphe (partiel) commun de G et G' à un cardinal (nombre de noeuds) inférieur.

Classification en termes de graphes

- L'appariement non bijectif :
  - Trouver $\phi : V' \rightarrow \mathcal{P}(V)$ qui associe à chaque sommet du graphe modèle $v' \in V'$ un ensemble de sommets du graphe scène et tel que :

1. chaque sommet de V est associé à exactement 1 sommet de V'.
2. $v \in \phi(v')$ ssi cet appariement est vraisemblable (en terme de similarité)
3. chaque sous graphe de G induit par $\phi(v'), v' \in V'$ est connexe.
4. pour tout $v' \in V', |\phi(v')| \geq 1$.

Applications aux images sur segmentées.

⚠ fondamental en terme d'applications.

Isomorphisme de graphes ou de sous graphes :

- Isomorphisme de graphes ou de sous graphes : Problème NP complet
- Heuristiques pour obtenir des solutions (éventuellement) sous optimales.
- Deux approches :
- approche symbolique ou algorithmique :
    - Définir un algorithme qui éffectue une recherche dans l'espace des solutions en rejetant à priori certaines solutions.
    - Bon contrôle sur la solution.
- approche numérique :
    - Définir le problème en terme de minimisation/maximisation d'une fonction/énergie.
    - Utilisation de tout l'arsenal des méthodes de minimisation

Principaux Acteurs

- Approches Algorithmiques
  - Horst Bunke (Suisse),
  - Marcello Pellilo(Italie),
  - Mario Vento (Italie).

- Approches numériques
  - Edwin Hancock (UK),
  - Kittler (UK),
  - Sven Dickinson (Canada),

Isomorphisme

- Matrice d'adjacence et permutation (Ullman Bunke)
- Appariement par représentation de l'espace des états
- Algorithme Hongrois (algorithme de Kühn), minimisation d'affectation.

Approches algorithmiques (Ullman/Bunke)

- Définitions :
  - Un **graphe labélisé** :$G = (V, E, \mu, \nu, L_v, L_e)$

$$\begin{cases} \mu & : V \rightarrow L_v & \text{fonction de label des sommets} \\ \nu & : E \rightarrow L_e & \text{fonction de label des arêtes} \end{cases}$$

  - Un **sous graphe labélisé** $G_s = (V_s, E_s, \mu_s, \nu_s, L_v, L_e)$ de $G = (V, E, \mu, \nu, L_v, L_e)$.
    - μ_s et ν_s restriction de μ et ν à $V_s \subset V$ et $E_e \subset E$ (avec $E_s = E \cap V_s \times V_s$).

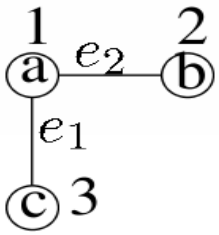
Matrice d'adjacence, permutation

- La **matrice d'adjacence** $M = (m_{i,j})$ d'un graphe $G = (V, E, \mu, \nu, L_v, L_e$ définie par :
  1. $\forall i \in \{1, \ldots, n\}\ m_{i,i} = \mu(v_i)$
  2. $\forall (i,j) \in \{1, \ldots, n\}^2, i \neq j$

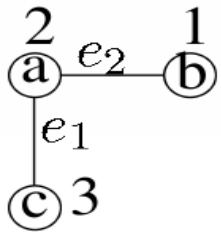
$$m_{i,j} = \begin{cases} \nu((v_i, v_j)) & \text{si } (v_i, v_j) \in E \\ 0 & \text{sinon} \end{cases}$$

- Une **matrice de permutation** $n \times n$, $P = (p_{i,j})$ vérifie :
  1. $\forall (i,j) \in \{1, \ldots, n\}^2\ p_{i,j} \in \{0, 1\}$,
  2. $\forall j \in \{1, \ldots, n\}\ \sum_{i=0}^{n} p_{i,j} = 1$,
  3. $\forall i \in \{1, \ldots, n\}\ \sum_{j=0}^{n} p_{i,j} = 1$.

Matrice d'adjacence, Permutation



ou



$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} a & e_2 & e_1 \\ e_2 & b & 0 \\ e_1 & 0 & c \end{pmatrix} \end{array} , \quad P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

$$M' = PMP^t = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} b & e_2 & 0 \\ e_2 & a & \text{e1} \\ 0 & \text{e1} & c \end{pmatrix} \end{array}$$

Bunke/Ullman : Définitions

- Deux graphes G_1 et G_2 de matrices M_1 et M_2 sont dis **isomorphes** ssi il existe P matrice de permutation telle que :

$$M_2 = PM_1P^t$$

- Il existe un isomorphisme de sous graphe entre G_1 et G_2 ssi il existe $S \subset G_2$ tel que G_1 et S sont isomorphe $(S = (S, E \cap S \times S, \mu_{|S}, \nu_{|S}, L_v, L_e)$.
- Soit $M = (m_{i,j})$ une $n \times n$ matrice d'adjacence.

$$\forall (k,m) \in \{1,\ldots,n\}^2 \; S_{k,m}(M) = (m_{i,j})_{i \in \{1,\ldots,k\}, j \in \{1,\ldots,m\}}$$

  - $S_{k,k}(M)$ matrice d'adjacence du sous graphe restreint aux k premiers sommets.

Bunke/Ullman : isomorphisme de sous-graphe

- Soient G_1 et G_2 de matrices d'adjacences M_1 et M_2
  - $M_1 : m \times m$,
  - $M_2 : n \times n$ avec $m \leq n$.

  Il existe un isomorphisme de sous graphe entre G_1 et G_2 ssi il existe une matrice de permutation $n \times n$ P telle que :

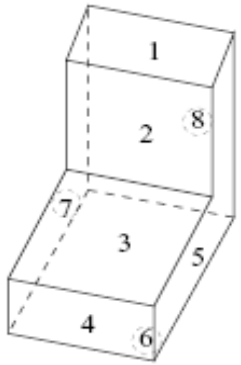
$$M_1 = S_{m,m}(PM_2P^t)$$

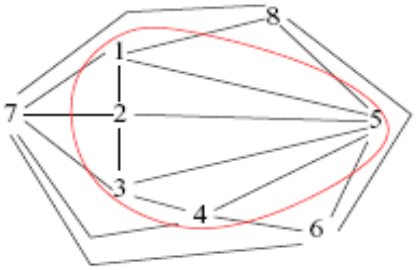
- Remarque 1 :

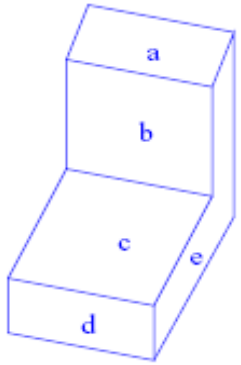
$$M_1 = S_{m,m}(PM_2P^t) = S_{m,n}(P)M_2\left(S_{m,n}(P)\right)^t$$

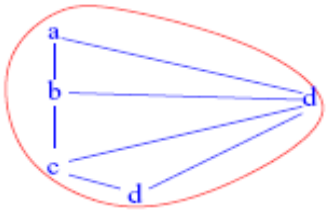
Isomorphismes de sous-graphes

- Il existe un sous-graphe S' de G' tel que f soit un isomorphisme de G dans S'









- Il existe un sous-graphe S de G et un sous-graphe S' de G' tel que f soit un isomorphisme de S dans S'

Bunke/Ullman : isomorphisme de sous-graphe

- Remarque 2 : si pour $i \leq m \leq n$

$$S_{i,i}(M_1) = S_{i,i}(PM_2P^t) = S_{i,n}(P)M_2\left(S_{i,n}(P)\right)^t$$

$S_{i,n}(P)$ représente un appariement partiel entre les i premiers sommets de G_1 et les sommets de G_2.

- L'algorithme

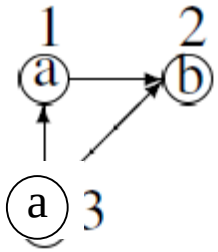
procédure Ullman (G_1,G_2)
  Déclaration
  $P = (p_{i,j})$ matrice $n \times n$ de permutation,
  $m = |V_1|$,$n = |V_2|$,
  M_1, M_2 matrices d'adjacences
début
  retourner backtrack($M_1, M_2, P, 1$)
fin

Bunke/Ullman : isomorphisme de sous-graphe

procédure backtrack (M_1, M_2, P, k)
début
  si $k > m$ **alors**
    Remarque : *P est un isomorphisme de sous graphe.*
    retourner P
  finsi
  pour i $\leftarrow$1 **à** n **faire**
    $p_{k,i} \leftarrow 1$
    $\forall j \neq i \; p_{k,j} \leftarrow 0$
  finpour
  si $S_{k,k}(M_1) = S_{k,n}(P) M_2 S_{k,n}(P)^t$ **alors**
    backtrack($M_1, M_2, P, k+1$)
  finsi
fin

Bunke: Exemple

- Étant donné une matrice M_i de G_i calculer l'ensemble $A(G_i)$ des graphes isomorphes à G_i :



1	2	3
b	1	1
0	a	1
0	0	a

A

1	3	2
b	1	1
0	a	0
0	1	a

B

2	1	3
a	0	1
1	b	1
0	1	a

C

2	3	1
a	1	0
0	a	0
1	1	a

D

3	1	2
a	0	0
1	b	1
1	0	a

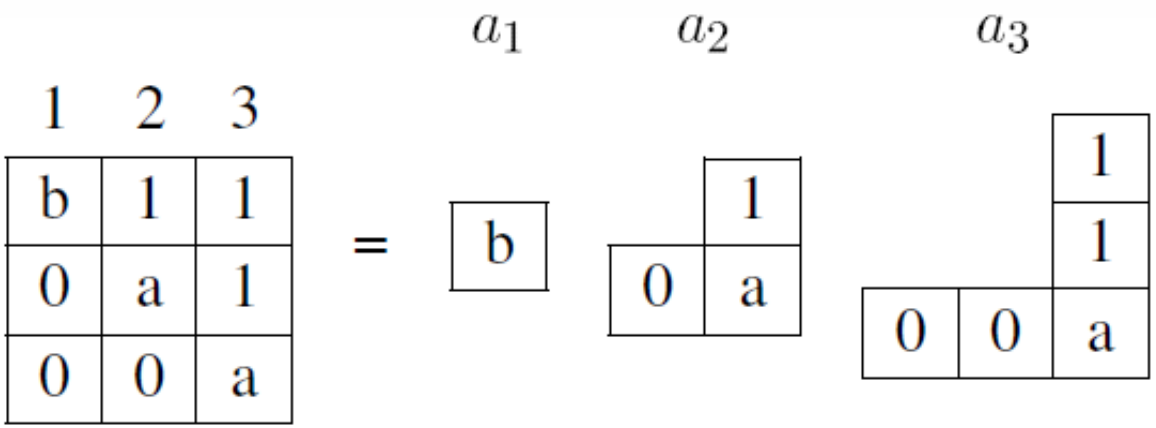
E

3	2	1
a	0	0
1	a	0
1	1	b

F

Bunke : Exemple

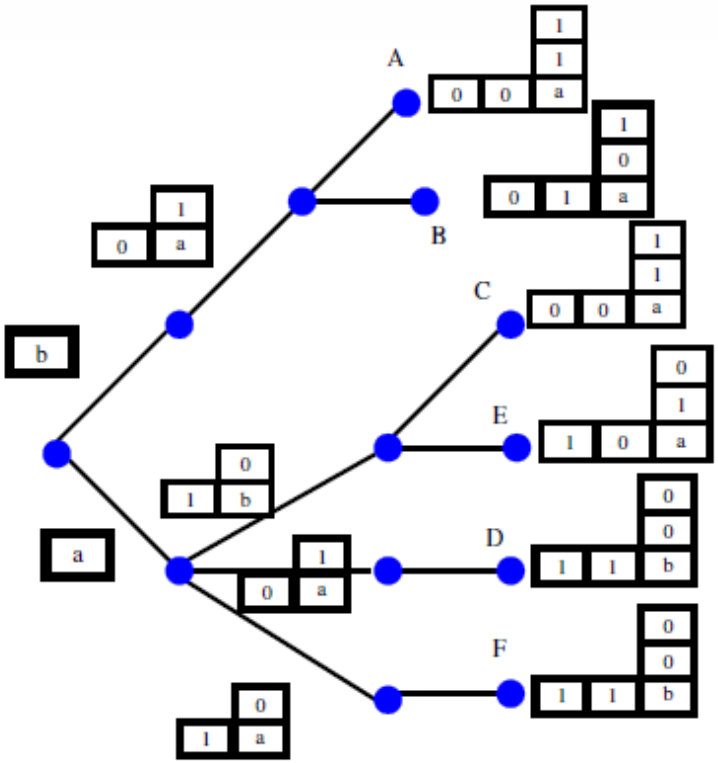
- Décomposer une matrice en vecteurs :



$$\begin{cases} a_1 = (b) \\ a_2 = (1, a, 0) \\ a_3 = (1, 1, a, 0, 0) \end{cases}$$

Bunke: Exemple

- Construire un arbre de recherche à partir de la décomposition



1	2	3
b	1	1
0	a	1
0	0	a

A

1	3	2
b	1	1
0	a	0
0	1	a

B

2	1	3
a	0	1
1	b	1
0	1	a

C

2	3	1
a	1	0
0	a	0
1	1	a

D

3	1	2
a	0	0
1	b	1
1	0	a

E

3	2	1
a	0	0
1	a	0
1	1	b

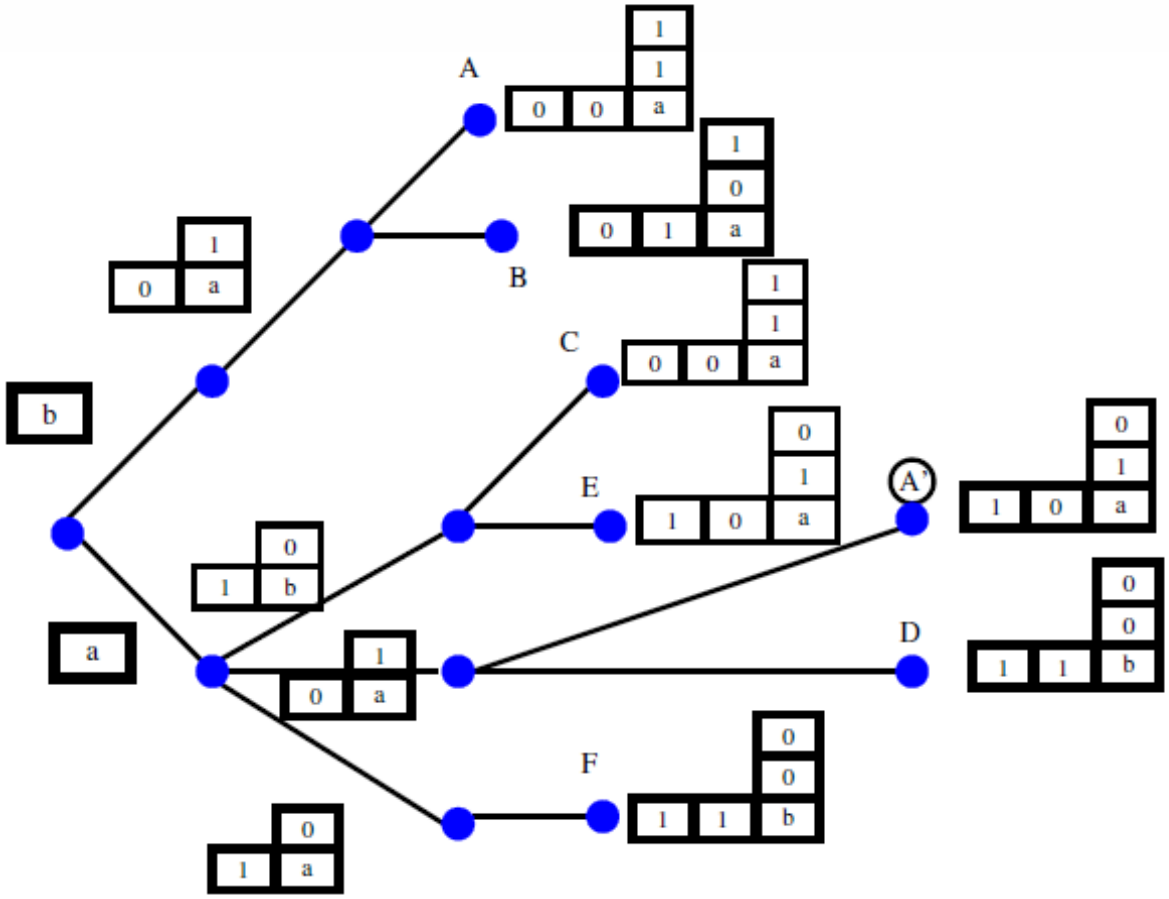
F

Bunke: Exemple

- Créer un arbre contenant tous les modèles.

1	2	3
a	1	0
0	a	1
1	0	a

A’



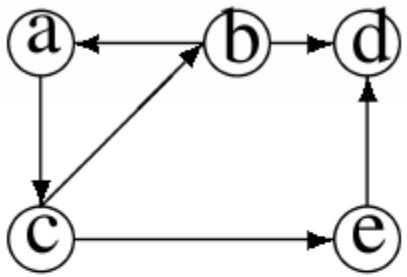
Appariement par représentation de l'espace des états

- State Space Representation (SSR).
- Un état : Un appariement partiel.
- Méthode : explorer successivement les différents états.
- Différentiation des méthodes :
  - Passage d'un état à un autre,
  - Méthode pour ne pas boucler (considérer 2 fois le même état),
  - Heuristique pour restreindre l'espace de recherche.

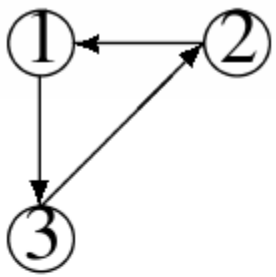
SSR : Algorithme de Cordella 2001-VF2

- Notations :
  - $G_1 = (V_2, E_2)$, $G_1 = (V_2, E_2)$ les deux graphes *orientés*,
  - On cherche soit :
    - un isomorphisme entre G_1 et G_2,
    - Soit un isomorphisme de sous graphe entre G_2 et G_1 ($|V_2| \leq |V_1|$)
  - état s,
  - $M(s)$ appariement partiel associé à s,
  - $M_1(s)$ sommets de $M(s)$ dans V_1,
  - $M_2(s)$ sommets de $M(s)$ dans V_2
  - $P(s)$ ensemble des couples (dans $V_1 \times V_2$) candidats à un ajout dans s,
  - $F(s, n, m)$ l'ajout de (n, m) à s définit il un isomorphisme partiel ?
  - $T_1^{in}(s)(T_1^{out}(s))$ ensemble des sommets de G_1 prédécesseurs (successeurs) d'un sommet de $M_1(s)$.
  - $T_2^{in}(s)(T_2^{out}(s))$ ensemble des sommets de G_2 prédécesseurs (successeurs) d'un sommet de $M_2(s)$.

Cordella : exemple de notations



G_1



G_2

- $M(s) = (a, 1)$
- $M_1(s) = \{a\}, M_2(s) = \{1\}$
- $T_1^{in}(s) = \{b\}; T_1^{out}(s) = \{c\};$
- $T_2^{in}(s) = \{2\}; T_2^{out}(s) = \{3\};$

Cordella : l'algorithme

procédure appariement (E s)
début
Remarque : s_0 *entrée initiale telle que* $M(s_0) = \emptyset$
si $M(s)$ contient tous les sommets de G_2 **alors**
retourner $M(s)$
sinon
Calculer $P(s)$
Pour chaque (n, m) **dans** $P(s)$ **faire**
si $F(s, n, m)$ **alors**
Calculer s' après ajout de (n, m) à $M(s)$
appariement(s')
finsi
finpour
Restauration des structures de données
finsi
fin

Calcul de *P(s)*

- Si $T_1^{out}(s)$ et $T_2^{out}(s)$ non vides

$$P(s) = T_1^{out}(s) \times \{minT_2^{out}(s)\}$$

  min relation d'ordre quelconque.
- Sinon Si $T_1^{in}(s)$ et $T_2^{in}(s)$ non vides

$$P(s) = T_1^{in}(s) \times \{minT_2^{in}(s)\}$$

- Sinon si $T_1^{in}(s) = T_2^{in}(s) = T_1^{out}(s) = T_2^{out}(s) = \emptyset$

$$P(s) = (V_1 - M_1(s)) \times \{min(V_2 - M_2(s))\}$$

- Note : Si un des $T^{in}(s)$ (resp. $T^{out}(s)$) est vide et pas l'autre $M(s)$ ne peut conduire à un appariement.

Calcul de *F(s,n,m)*

$$F(s,n,m) = R_{pred}(s,n,m) \wedge R_{succ}(s,n,m) \wedge R_{in}(s,n,m) \wedge R_{out}(s,n,m) \wedge R_{new}(s,n,m)$$

- R_{pred}, R_{succ} : M(s') définit il un appariement ?
- R_{in}, R_{out} : pourra t on construire un appariement au coup d'après ?
- R_{new} : Pourrai je arriver à un apariement (**à** terme) ?
  - $R_{pred}(s,m,n)$: les prédécesseurs correspondent :

$$(\forall n' \in M_1(s) \cap Pred(G_1,n) \exists m' \in Pred(G_2,m) | (n',m') \in M(s)) \wedge$$
$$(\forall m' \in M_2(s) \cap Pred(G_2,m) \exists n' \in Pred(G_1,n) | (n',m') \in M(s))$$

  - $R_{succ}(s,m,n)$: les successeurs correspondent :

$$(\forall n' \in M_1(s) \cap Succ(G_1,n) \exists m' \in Succ(G_2,m) | (n',m') \in M(s)) \wedge$$
$$(\forall m' \in M_2(s) \cap Succ(G_2,m) \exists n' \in Succ(G_1,n) | (n',m') \in M(s))$$

Prédicats *Rin* et *Rout*

- Les successeurs (prédécesseurs) de n et m doivent être en correspondance localement.
  - $R_{in}(s, n, m)$

$$\left(\left|T_1^{in}(s) \cap Succ(G_1, n)\right| \geq \left|T_2^{in}(s) \cap Succ(G_2, m)\right|\right) \wedge$$
$$\left(\left|T_1^{in}(s) \cap Pred(G_1, n)\right| \geq \left|T_2^{in}(s) \cap Pred(G_2, m)\right|\right)$$

  - $R_{out}(s, n, m)$

$$\left(\left|T_1^{out}(s) \cap Succ(G_1, n)\right| \geq \left|T_2^{out}(s) \cap Succ(G_2, m)\right|\right) \wedge$$
$$\left(\left|T_1^{out}(s) \cap Pred(G_1, n)\right| \geq \left|T_2^{out}(s) \cap Pred(G_2, m)\right|\right)$$

- Remplacer $\geq$ par $=$ pour l'isomorphisme de graphe.

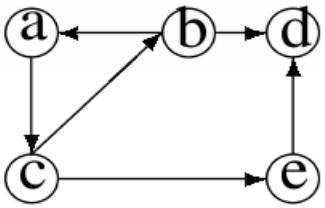
Prédicat *Rnew*

- Les successeurs et prédécesseurs doivent se correspondre en dehors des $M_i(s), T_i^{in}(s)$ et $T_i^{out}(s)$, $i = 1, 2$.
  - $R_{new}(s, n, m)$

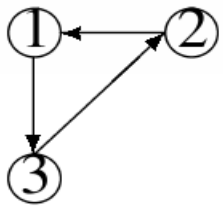
$$(|N_1(s) \cap Succ(G_1, n)| \geq |N_2(s) \cap Succ(G_2, m)|) \wedge$$
$$(|N_1(s) \cap Pred(G_1, n)| \geq |N_2(s) \cap Pred(G_2, m)|)$$

  - avec :
    - $N_1(s) = V_1 - M_1(s) - T_1^{in}(s) \cup T_1^{out}(s)$: tout ce qui reste a voir dans V_1
    - $N_2(s) = V_2 - M_2(s) - T_2^{in}(s) \cup T_2^{out}(s)$: tout ce qui reste a voir dans V_2

Exemple



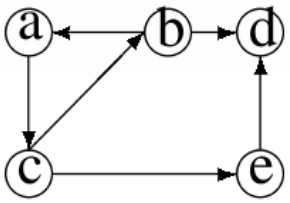
G_1



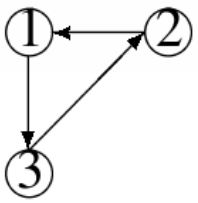
G_2

- $M(s) = (a, 1)$
- $M_1(s) = \{a\}, M_2(s) = \{1\}$
- $T_1^{in}(s) = \{b\}; T_1^{out}(s) = \{c\};$
- $T_2^{in}(s) = \{2\}; T_2^{out}(s) = \{3\};$
- $P(s) = (c, 3)$
- $Pred(G_1, c) = \{a\}; Succ(G_1, c) = \{b, e\}$
- $Pred(G_2, 3) = \{1\}; Succ(G_2, 3) = \{2\}$
- $F(s, c, 3) = vrai.$

Exemple



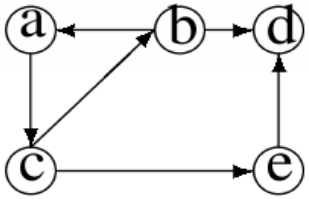
G_1



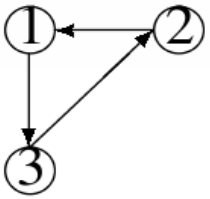
G_2

- $M(s') = \{(a,1),(c,3)\}$
- $M_1(s') = \{a,c\}, M_2(s) = \{1,3\}$
- $T_1^{in}(s') = \{b\}; T_1^{out}(s') = \{b,e\};$
- $T_2^{in}(s') = \{2\}; T_2^{out}(s') = \{2\};$
- $P(s') = (b,2),(e,2)$
- $Pred(G_1,e) = \{c\}; Succ(G_1,e) = \{d\}$
- $Pred(G_2,2) = \{3\}; Succ(G_2,2) = \{1\}$
- $(e,2)$ viole le prédicat $R_{succ}(s',e,2)$. De fait :
  - $1 \in M_2(s') \cap Succ(G_2,2)$ or $Succ(G_1,e) \cap M_1(s') = \emptyset$

Exemple



G_1



G_2

- On arrive donc à l'appriement : $M(s'') = \{(a, 1), (c, 3), (b, 2)\}$ et l'on a terminé puisque l'on couvre tous les sommets de G_2.

Conclusion

- Complexité :($N = |V_1| + |V_2|$)

	VF2		Ullman	
Complexité	au mieux	au pire	au mieux	au pire
Temps	$\mathcal{O}(N^2)$	$\mathcal{O}(N!N)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N!N^2)$
Espace	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N^3)$

- Utilisable pour de gros graphes (jusqu'a 1000 sommets).

Algorithme Hongrois

Permet de résoudre des problèmes d'affectation.

On représente dans une matrice le coût de l'affectation entre deux éléments (ce coût peut être dans notre cas un critère de ressemblance)

On calcule alors le coût minimal

Exemple

Réduction du tableau initial: on soustrait à chaque ligne du tableau initial le plus petit élément de la ligne. On fait de même avec les colonnes.

17	**15**	**9**	**5**	**12**
16	16	10	5	10
12	15	14	11	5
4	8	14	17	13
13	9	8	12	17

12	**10**	**4**	**0**	**7**
11	11	5	0	5
7	10	9	6	0
0	4	10	13	9
5	1	0	4	9

12	**9**	**4**	**0**	**7**
11	10	5	0	5
7	9	9	6	0
0	3	10	13	9
5	0	0	4	9

Exemple

Encadrer et barrer des zéros

On cherche la ligne comportant le moins de zéros non barrés (en cas d'égalité, choisir la ligne la plus haute). On encadre un des zéros de cette ligne (arbitrairement le plus à gauche)

On barre tous les zéros se trouvant sur la même ligne ou sur la même colonne que le zéro encadré.

12	**9**	**4**	**0**	**7**
11	10	5	~~0~~	5
7	9	9	6	0
0	3	10	13	9
5	0	0	4	9

12	**9**	**4**	**0**	**7**
11	10	5	~~0~~	5
7	9	9	6	0
0	3	10	13	9
5	0	0	4	9

Exemple

12	**9**	**4**	**0**	**7**
11	10	5	~~0~~	5
7	9	9	6	0
0	3	10	13	9
5	0	~~0~~	4	9

			x		
12	**9**	**4**	**0**	**7**	x
11	10	5	~~0~~	5	x
7	9	9	6	0	
0	3	10	13	9	
5	0	~~0~~	4	9	

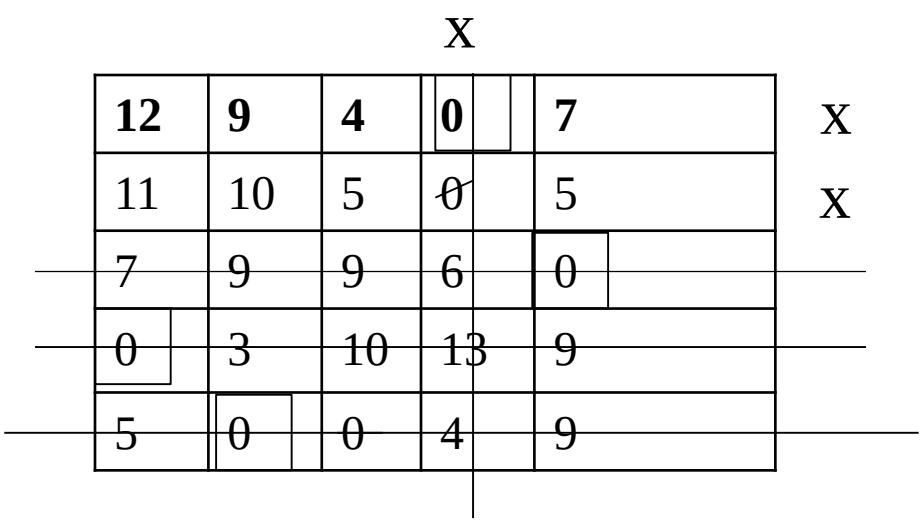
1- Marquer d'une croix toutes les lignes ne contenant aucun zéro encadré

2- Marquer toute colonne ayant un zéro barré sur une ligne marquée

3- Marquer toute ligne ayant un zéro encadré dans une colonne marquée

On répète 2 et 3 jusqu'à ne plus pouvoir marquer de ligne ou de colonne

Exemple



12	**9**	**4**	**0**	**7**
11	10	5	~~0~~	5
7	9	9	6	0
0	3	10	13	9
5	0	0	4	9

12	**9**	**4**	**0**	**7**
11	10	5	~~0~~	5
7	9	9	6	0
0	3	10	13	9
5	0	~~0~~	4	9

- On trace un trait sur toute ligne non marquée et sur toute colonne marquée
- On retranche à toutes les cases du tableau partiel le plus petit élément de celui-ci
- On ajoute ce même élément à toutes les case du tableau initial barrées deux fois

Exemple

12	**9**	**4**	**0**	**7**
11	10	5	~~0~~	5
7	9	9	6	0
0	3	10	13	9
5	0	~~0~~	4	9

8	**5**	**0**	**0**	**3**
7	6	1	0	1
7	9	9	10	0
0	3	10	17	9
5	0	~~0~~	8	9

On obtient un nouveau tableau sur lequel on peut répéter les trois étapes

En revenant sur le tableau initial on obtient la valeur de l'affectation minimale 9+5+5+4+9=32

Algorithme hongrois

Ce principe peut être utilisé pour effectuer l'appariement des nœuds d'un graphe.

Appariement d'ensembles avec édition, Application à la distance d'édition bipartie entre graphes,

Sébastien Bougleux, Luc Brun, Benoit Gaüzère, RFIA 2016, Clermont ferrand.

Problèmes de sous graphes communs

- Liens entre les isomorphismes de sous graphes et les sous graphes communs
  - Étant donné deux graphes G_1 et G_2, G est un sous graphe commun de G_1 et G_2 ssi il existe :
    - $G \xrightarrow{\varphi} G_1$
    - $G \xrightarrow{\psi} G_2$

    φ, ψ isomorphismes de sous graphes.
- G est maximum (maximal) si on ne peut trouver de sous graphe de cardinal supérieur (incluant G).
- Première idée : utiliser un algorithme SSR (VF2,McGregor).

Utilisation d'algorithmes SSR

procédure appariement (E s)
début
Remarque : s_0 *entrée initiale telle que* $M(s_0) = MCS = \emptyset$
si $M(s)$ contient tous les sommets de G_2 **alors**
retourner $M(s)$
sinon
Calculer $P(s)$
Pour chaque (n, m) **dans** $P(s)$ **faire**
si $F(s, n, m)$ **alors**
Calculer s' après ajout de (n, m) à $M(s)$
si taille(M(s'))> tailleMax **alors**
tailleMax ← taille(M(s'))
MCS ← M(s')
finsi
appariement(s')
finsi

Graphe d'association

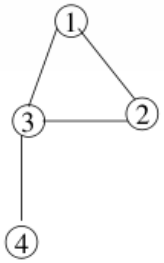
- Soient $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$, le graphe d'association $G = (V, E)$ de G_1 et G_2 est défini par :
  - Sommets :

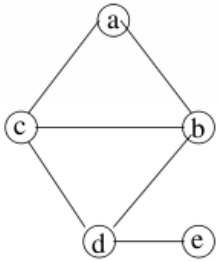
$$V = V_1 \times V_2$$

  - Arêtes :

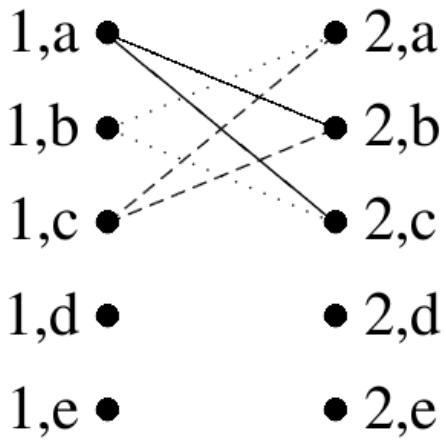
$$E = \{((i,h),(j,k)) \in V \times V | i \neq j, h \neq k \text{ et } (i,j) \in E_1 \Leftrightarrow (h,k) \in E_2 \}$$

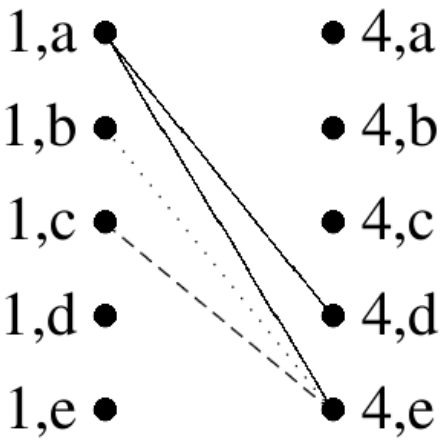
Graphe d'association : exemple



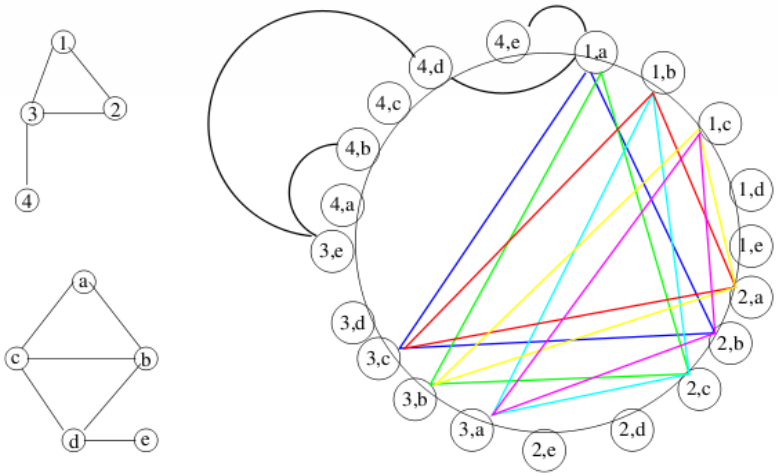


mais également





Graphe d'association : exemple



- Le sous graphe de G restreint à $\{(1,a),(2,b),(3,c)\}$ est complet (lignes bleu)
  - Tous les appariements sont compatibles.

MCS et clique

- Un sous graphe complet d'un graphe G est appellé un **clique** de G.
- On parle de clique maximal (resp. maximum).
- Le clique-number de G, w(G), est la taille (en nombre de sommets) du clique maximum.
- Théorème :

Soient G1 et G2 deux graphes et G leur graphe d'association. Il existe une relation bijective entre

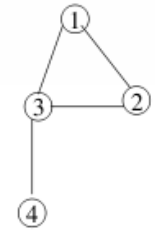
- *les cliques maximal (maximum) de G et*
- *les sous graphes communs maximal (maximum) de G1 et G2.*

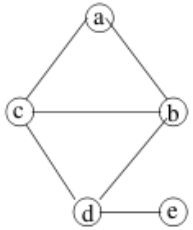
- Donc : Calculer les cliques du graphe d'association G est **équivalent** à calculer les sous graphes communs de G1 et G2.

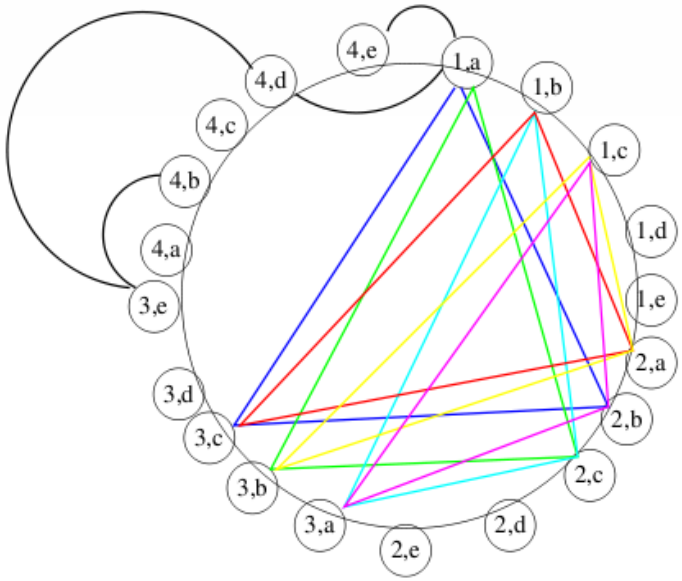
DurandParisi (1999)

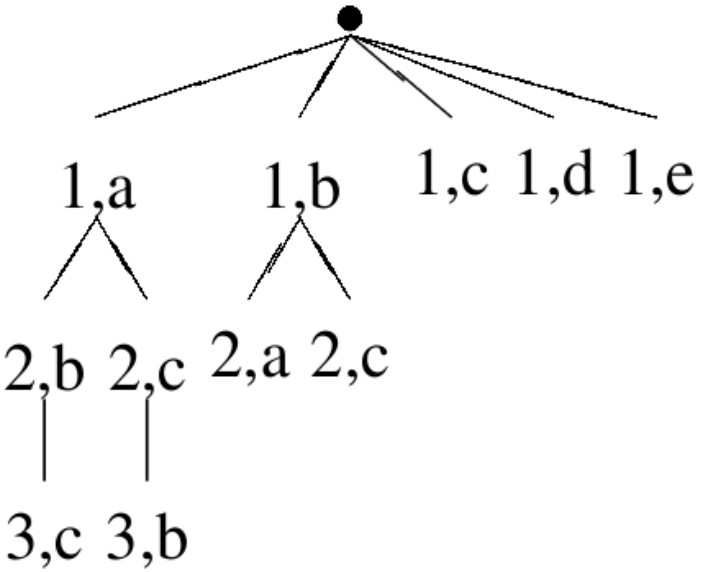
```
procédure durandParisi (E s)
début
    tant que nextNode(s,n) faire
        si isLegalNode(s,n) et non pruningCondition(s) alors
            s' ← addNode(s,n)
            si taille(s')> tailleMax alors
                MCS ← M(s')
                tailleMax ← taille
            finsi
            si non leafOfSearchTree(s') alors
                durandParisi(s')
            finsi
            backtrack(s')
        finsi
    fintantque
fin
```

DurandParisi : exemple









Plan du cours

- Pourquoi les graphes ?
- Rappels et définitions
- De l'image à l 'objet graphe: les graphes d'adjacence de régions
- Simplification de graphes
  - division et fusion de régions
  - fusion pyramidale
- Interprétation d 'images: Mise en correspondance de graphes
  - Isomorphisme de graphes et sous graphes
  - Isomorphisme de sous graphes avec tolérance d 'erreur
  - Mise en correspondance inexacte: un problème de satisfaction de contraintes

Isomorphismes de sous-graphes avec tolérance d'erreurs

- Monde réel : graphes bruités, incomplets, distorsions
- Distance entre graphes (opérations d'édition, fonction de coût,...)
- Isomorphisme de sous-graphe avec tolérance d'erreurs : recherche du sous-graphe de G' à distance minimale de G
- Algorithmes optimaux : A* (solution correcte assurée, complexité exponentielle)
- Algorithmes approximatifs : génétiques, recuit simulé, réseaux de neurones,relaxation probabiliste,...
  - minimisation itérative d'une fonction objectif (distance de la solution courante à la solution optimale)
  - mieux adaptés pour les grands graphes
  - problèmes de minima locaux et de convergence

Algorithme A*

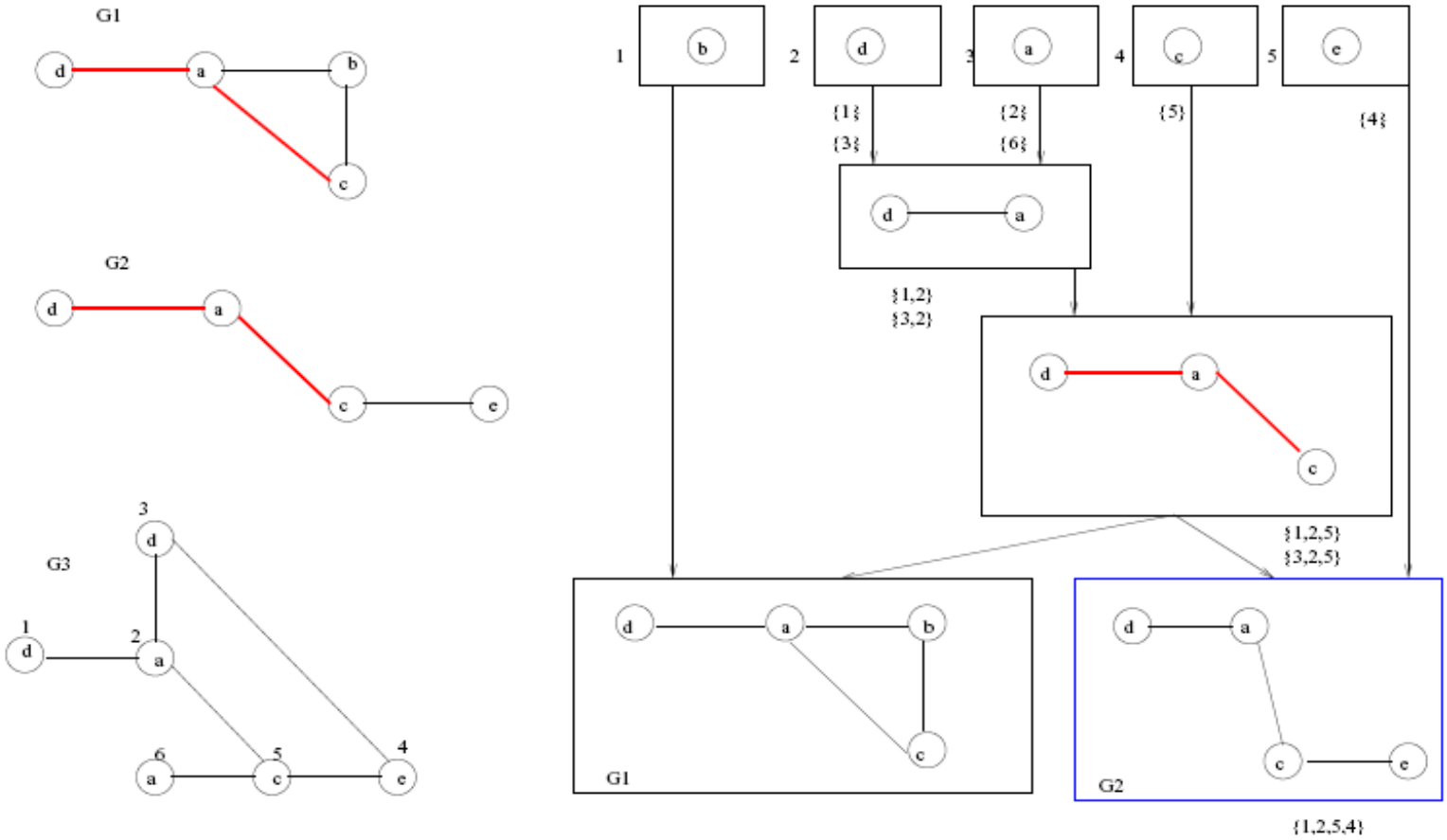
- Principe : construction d'un arbre de recherche par appariements successifs des sommets avec évaluation de la fonction de coût pour chaque état (seuls les états de coûts inférieurs sont ensuite propagés)
  - Complexité : $O(n^2 m^n)$ (pire cas)
  - Amélioration : estimation des coûts futurs pour ne pas propager inutilement des branches

Décomposition en éléments communs (Messmer et Bunke, PAMI 1998)

- Problématique: Soient un ensemble de graphes modèles (G1, … Gn) et un graphe en entrée G. Trouver la mise en correspondance la plus adaptée entre l ’un ces graphes modèles et le graphe G.
- Principe:
  - création préalable d ’une structure contenant la décomposition en éléments commun des graphes modèles.
  - Recherche des isomorphismes *ec* (error-correcting) de sous-graphe entre le graphe d ’entrée et les sous-graphes des graphes modèles (mise en correspondance par édition de graphe).
  - La meilleure mise en correspondance correspond à celle qui aura la plus petite distance d ’édition de graphe (isomorphisme *oec* de sous-graphe)

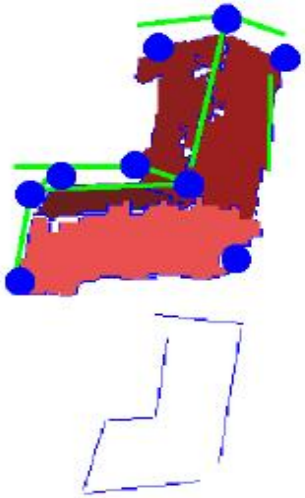
Décomposition en éléments communs

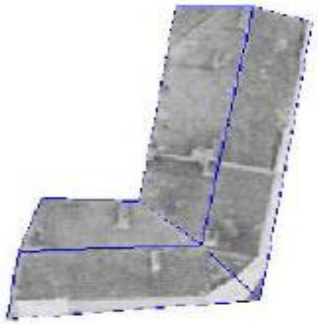
Messmer (1996)



Exemple

Reconstruction 3D par appariement entre un ensemble de graphes modèles et un graphe de données (IGN)





Optimisation

1- Représenter le graphe par une matrice

2- Transformer le problème de graphe en un problème de maximisation/minimisation d'une expression (en utilisant la matrice)

3- Choisir une méthode d'optimisation

Vecteur caractéristique

- Une forme est représentée par un vecteur de n composantes correspondant à la mesure de n caractéristiques observées sur elle.
- but : regrouper ces vecteurs en classes
  - Deux propriétés :
    - Compacité
    - Séparabilité
  - Notion de proximité donné par la distance

Et sur les graphes ...

- Nombre de noeuds, nombres d'arrêtes
- Nombre de noeuds avec le label A, ou le label B,...
- Nombre d'arrêtes entre le label A et le label C ,...degrée moyen des noeuds
- Nombre de cycles d'une certaines longueur

Et sur les graphes ...

La matrice d'adjacence de G_k est définie par

$$A_k(i, j) = \begin{cases} W(i, j), \text{ si } (i, j) \in E_k \\ 0, \text{ sinon} \end{cases}$$

Les valeurs propres λ_k de A_k sont les solutions de $|A_k - \lambda_k I| = 0$

On peut considérer toutes les valeurs propres comme des caractéristiques pour la représentation vectorielle de G_k : $B_k = (\lambda_k^1, \lambda_k^2, \ldots, \lambda_k^n)^T$

Les vecteurs propres φw sont les solutions de $A_k \varphi_k^w = \lambda_k \varphi_k^w$, ou w est l'index de mode propre.

La matrice modale est définie par $\varphi_k = (\varphi_k^1 \mid \varphi_k^2 \mid \ldots \mid \varphi_k^{|Vk|})$

La décomposition spectrale de la matrice d'adjacence est $Ak = \sum_{i=1}^{|Vk|} \lambda_k^w \varphi_k^w (\varphi_k^w)^T$

La matrice modale tronquée est définie par $\varphi k = (\varphi_k^1 \mid \varphi_k^2 \mid \ldots \mid \varphi_k^n)$

Vecteur caractéristique

Sur un graphe: G=(V,E) et C $\subset$ V et |V|=n

Vecteur caractéristique de C:

$$x^C = \begin{pmatrix} x_1 \\ . \\ . \\ . \\ x_n \end{pmatrix} \text{ avec } x_i = \begin{cases} \frac{1}{|C|} si \;\; i \in C \\ 0 \;\; sinon \end{cases}$$

Fonction Quadratique

- Soit $A_G=(a_{i,j})$ la matrice d'adjacence d'un graphe G
- et la fonction :

$$g(x)= x^t A_G x + \tfrac{1}{2} x^t x= x^t A x \text{ avec } \begin{cases} A = A_G + 1/2I \\ x \in Sn \end{cases}$$

x* est un maxima strict de g ssi:

$$\exists \varepsilon > 0 \mid \begin{cases} |y - x*| < \varepsilon \rightarrow g(y) \leq g(x*) \\ |y - x*| < \varepsilon \; et \; g(y) = g(x*) \rightarrow y = x* \end{cases}$$

Application à la reconnaissance d'objets

Représenter chaque objet par un graphe, par exemple un schock tree A = (V, E)

- Un arbre enraciné représentant le squelette de l'objet.
- Feuilles: détails de l'objets

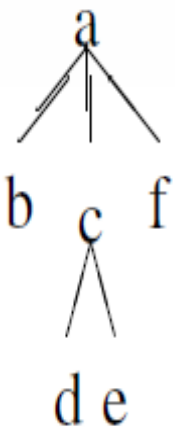
On peut définir un chemin entre deux sommets unique:

$\forall (u, v) \in V^2 \; \exists \; P = x_0, \ldots, x_n \; avec \; x_0 = u \; et \; x_n = v$ et $\forall i \in \{1, \ldots, n\}$ niveau(x_i) = niveau$(x_{i_1}) \pm 1$

Chaîne entre sommets:

Str(u,v)= $s_1, \ldots, s_n$ avec s_i=niveau(x_i) – niveau$(x_{i-1}) \in \{-1, 1\}$

Graphe d'association

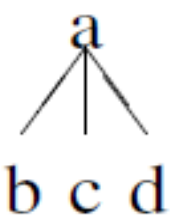


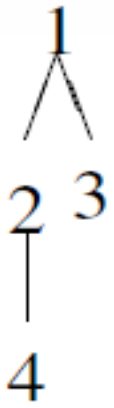
$$str(b, e) = 1. - 1. - 1$$

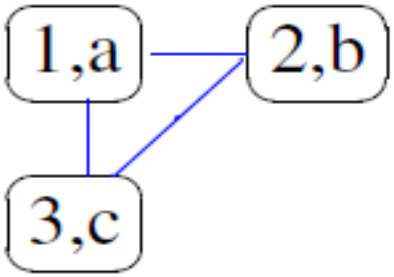
- Soient $A_1 = (V_1, E_1)$ et $A_2 = (V_2, E_2)$.
- On construit le graphe d'association $G = (V_1 \times V_2, E)$ avec :

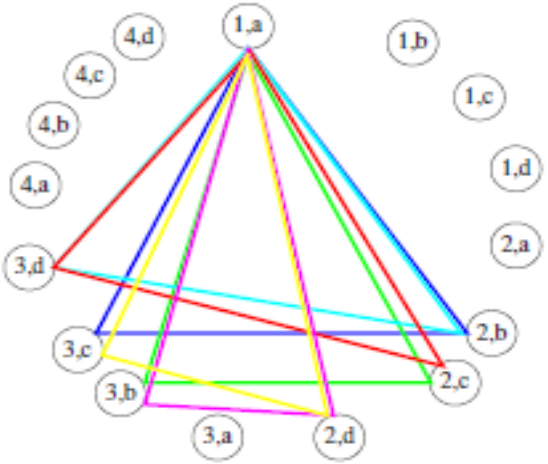
$$((u, w), (v, z)) \in E \Leftrightarrow Str(u, v) = Str(w, z)$$

Graphe d'association: exemple









Résolution

- Recherche des maxima de

$$g(x) = x^t Ax \text{ avec } A = A_G + \frac{1}{2} I$$

- Équations de replication :

$$x_i(t+1) = x_i(t) \frac{(Ax(t))_i}{g(x)} \text{ note } \sum_{i=1}^{n} x_i(t) = \frac{g(x)}{g(x)} = 1$$

- Converge vers un point stationnaire($x_i(t+1) = x_i(t)$).

Méthodes structurelles relaxation continue

- Méthode de Zucker et Hummel (1976): Introduction de probabilités pour exprimer les compatibilités.
- À chaque étiquette sur chaque nœud du graphe on associe un couple de probabilité (p,q)
  - p: importance de l'étiquette par rapport aux autres étiquettes du même noeud
  - q: degrés de cohérence de l'étiquette par rapport à toutes les étiquettes des noeuds voisins
- On ajuste les probabilités par propagation

Mise en correspondance
Approche par relaxation continue

- **But** : apparier les noeuds selon leur coefficient de similarité

- **Principe** :

Gomila, Meyer : *Graph-based object tracking*, ICIP 2003, 14-17 Sept., Princeton, USA

Chevalier, Domenger, Benois-Pineau, Delest : *Recognition of objects in video by simiilarity based on graph matching (2005)*

Mise en correspondance
Approche par relaxation continue

Mise en correspondance
Approche par relaxation contnue

- Similarité de couleur (distance RGB)

- Similarité de forme (distance descripteurs de forme)
  - Caractéristiques de la boite englobante orientée
  - Aire relative à l'objet

Mise en correspondance
Approche par relaxation continue

- Similarité de couleur $S_c(r,r')$

- Similarité de forme $S_s(r,r')$

- Similarité globale (dans l'intervalle [0;1])

$$S(r,r') = S_c(r,r').\ S_s(r,r')$$

Mise en correspondance
Approche par relaxation continue

Mise en correspondance
Approche par relaxation continue

- Similarité entre deux régions et leur contexte: meilleur recouvrement possible

$$\boldsymbol{N}\,(r,r') = 1/M\,(\Sigma_{\forall j \in Nr'} \max_{\forall i \in Nr} S(i,j))$$

avec M cardinal de Nr'

Mise en correspondance
Approche par relaxation continue

- Propriété d'assignation

$$r \approx r' \Longleftrightarrow \begin{cases} \forall r_i \neq r \in N_r & S(r_i, r') < S(r, r') \\ \forall r'_j \neq r' \in N_{r'} & S(r, r'_j) < S(r, r') \\ S(r, r') > \psi_s \end{cases}$$

Mise en correspondance
Approche par relaxation continue

- Propriété d'assignation

$$r \approx r' \Longleftrightarrow \begin{cases} \forall r_i \neq r \in N_r & S(r_i, r') < S(r, r') \\ \forall r'_j \neq r' \in N_{r'} & S(r, r'_j) < S(r, r') \\ S(r, r') > P_{min} \end{cases}$$

- Assignation ferme

Si $S(r,r') = max_{\forall i \in Nr} S(i,r')$, $S(r,r') = max_{\forall j \in Nr'} S(r,j)$ et $S(r,r') > P_{min}$ on effectue l 'assignation ferme suivante:

$$S(r, r') = 1$$
$$\forall r_k \in N_r \; r_k \neq r, \quad S(r_k, r') = 0$$
$$\forall r'_l \in N_{r'}, r'_l \neq r', \quad S(r, r'_l) = 0$$

Mise en correspondance
Approche par relaxation continue

Mise en correspondance
Approche par relaxation continue

- Similarité entre deux régions et leur contexte **à l'itération i**
- Si $i=0$ $S^0(r,r')=S(r,r')$
- Si $i>0$ $S^{(i)}=S^{(i-1)}(r,r').\ Q^{(i-1)}(r,r')$

- ou Q est un facteur de pondération qui dépend de la cohérence du voisinage

Mise en correspondance
Approche par relaxation continue

- Récapitulatif

- Complexité $O(t(nn')^2)$

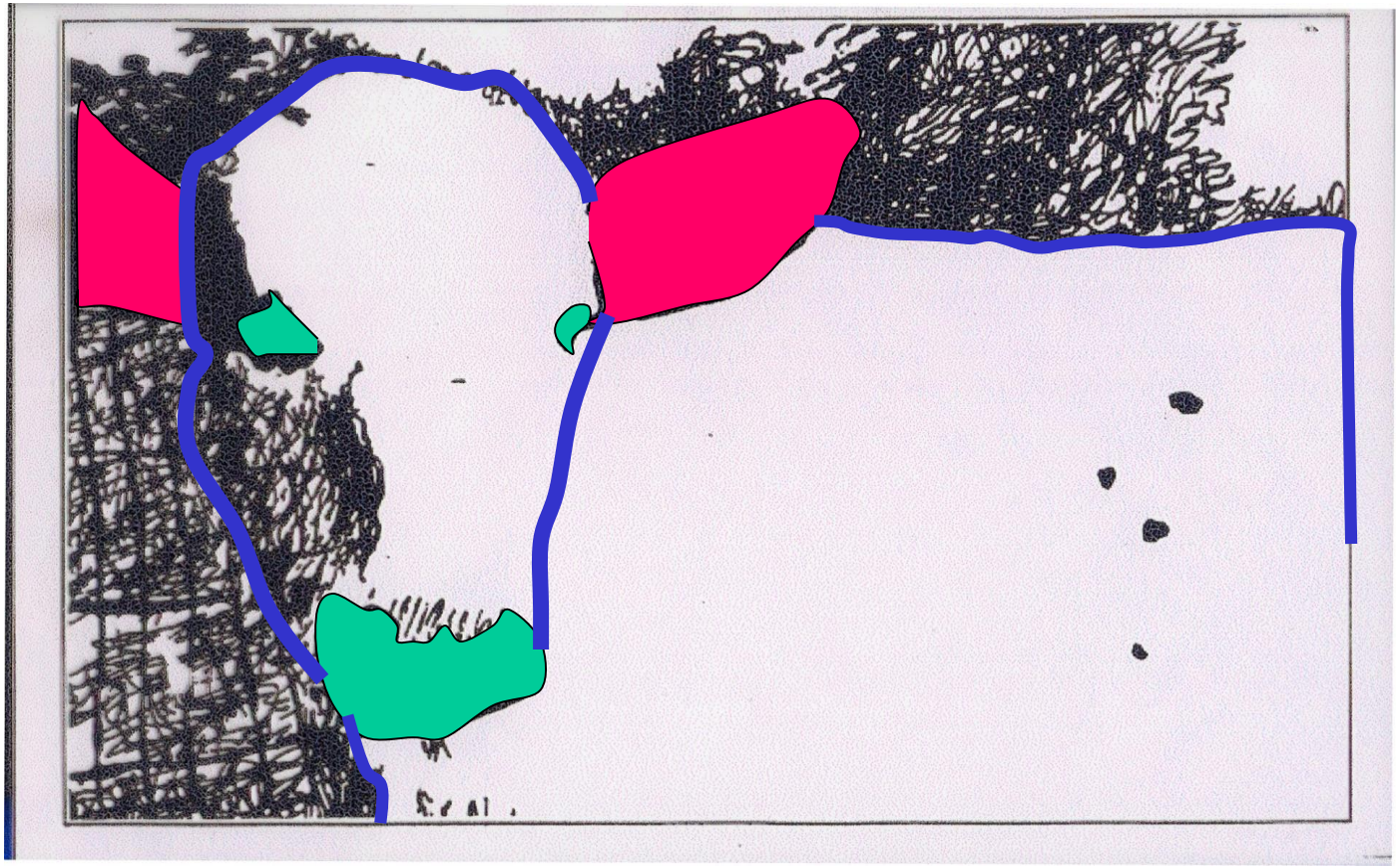
n (resp. n') : nombre de noeuds de G (resp. G')

t : nombre d'itérations

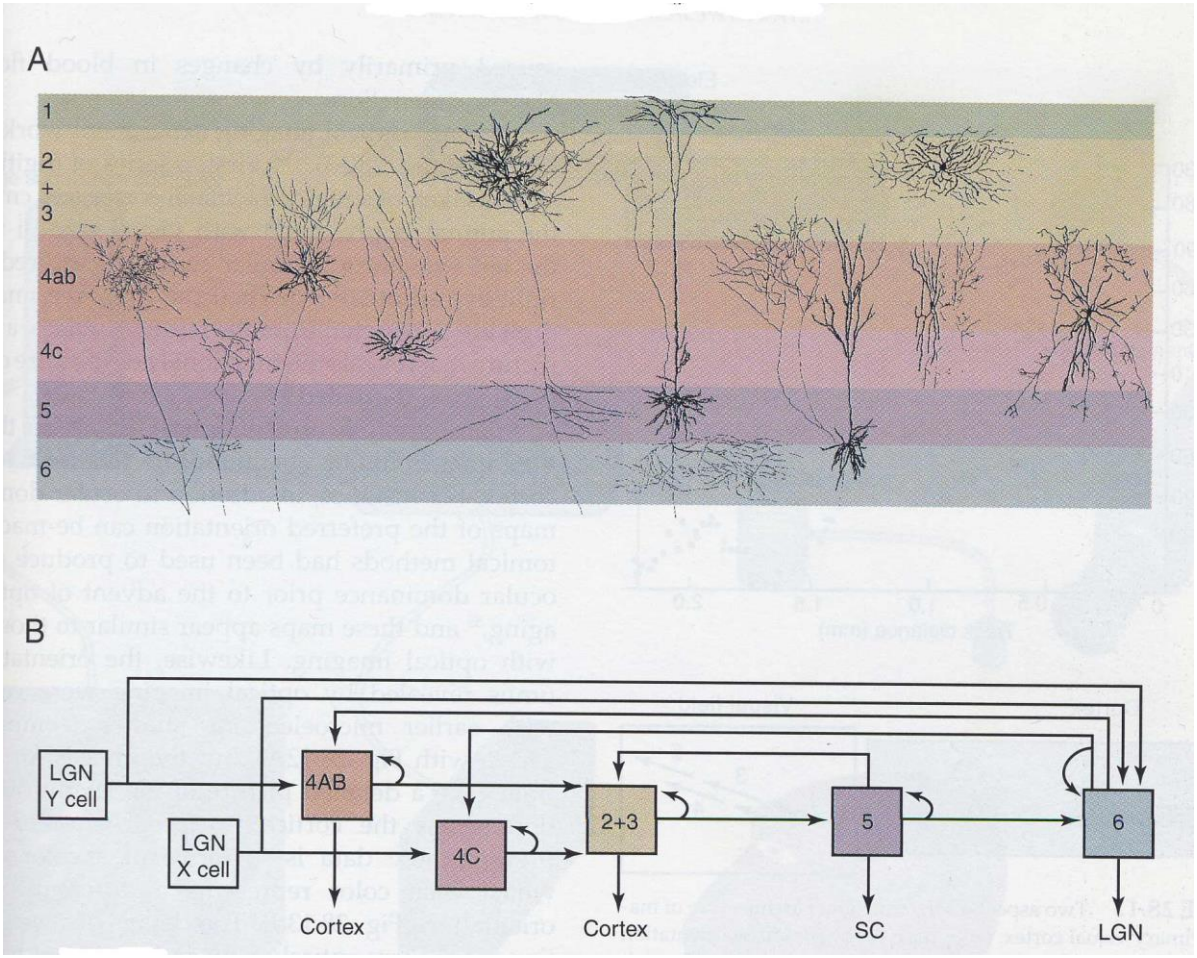
Plan du cours

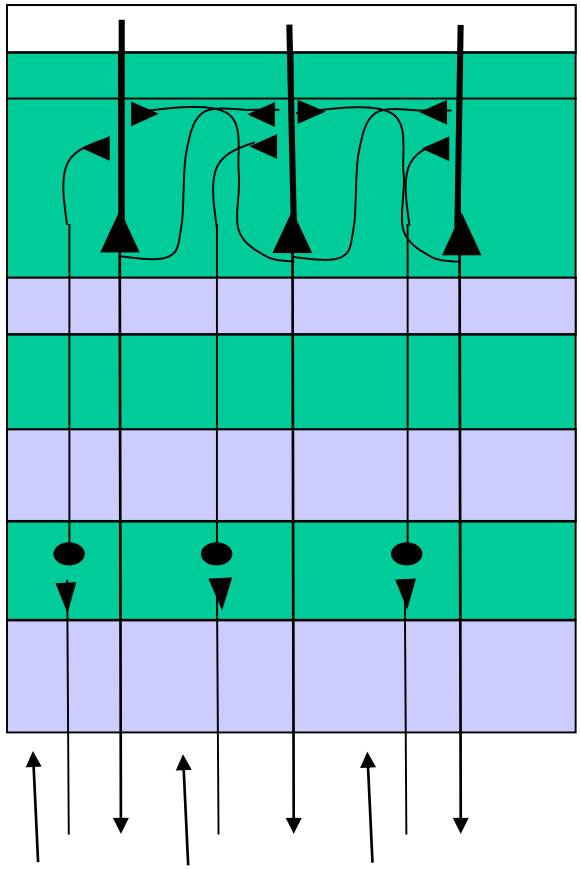
- Pourquoi les graphes ?
- Rappels et définitions
- De l'image à l 'objet graphe: les graphes d'adjacence de régions
- Simplification de graphes
  - division et fusion de régions
  - fusion pyramidale
- Interprétation d 'images: Mise en correspondance de graphes
  - Isomorphisme de graphes et sous graphes
  - Isomorphisme de sous-graphe avec tolérance d 'erreur
  - Mise en correspondance inexacte: un problème de satisfaction de contraintes

Relaxation discrète et interprétation d ’images

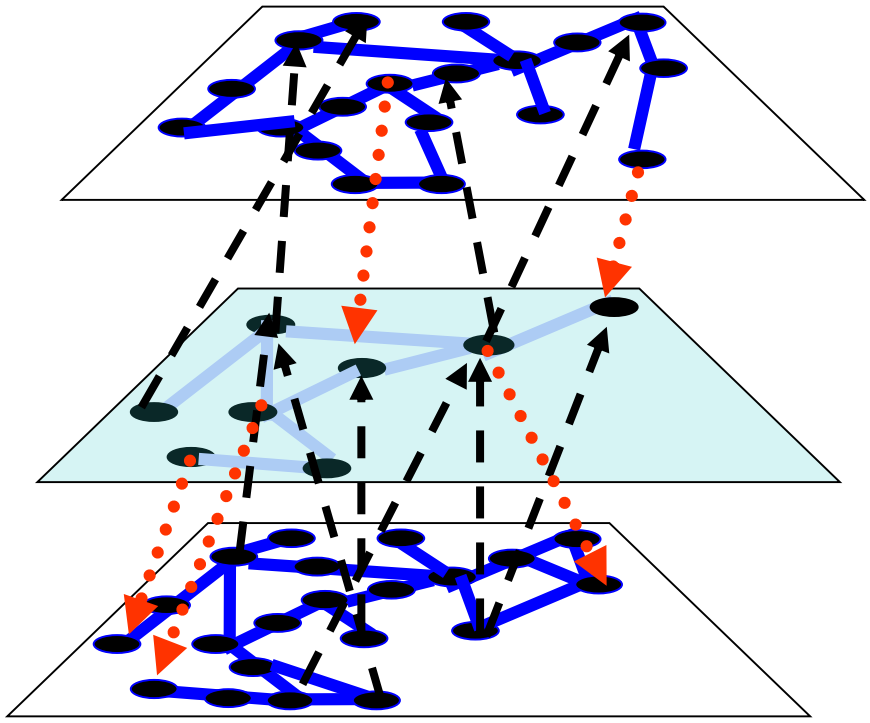


Le cortex visuel





Organisation neuronale

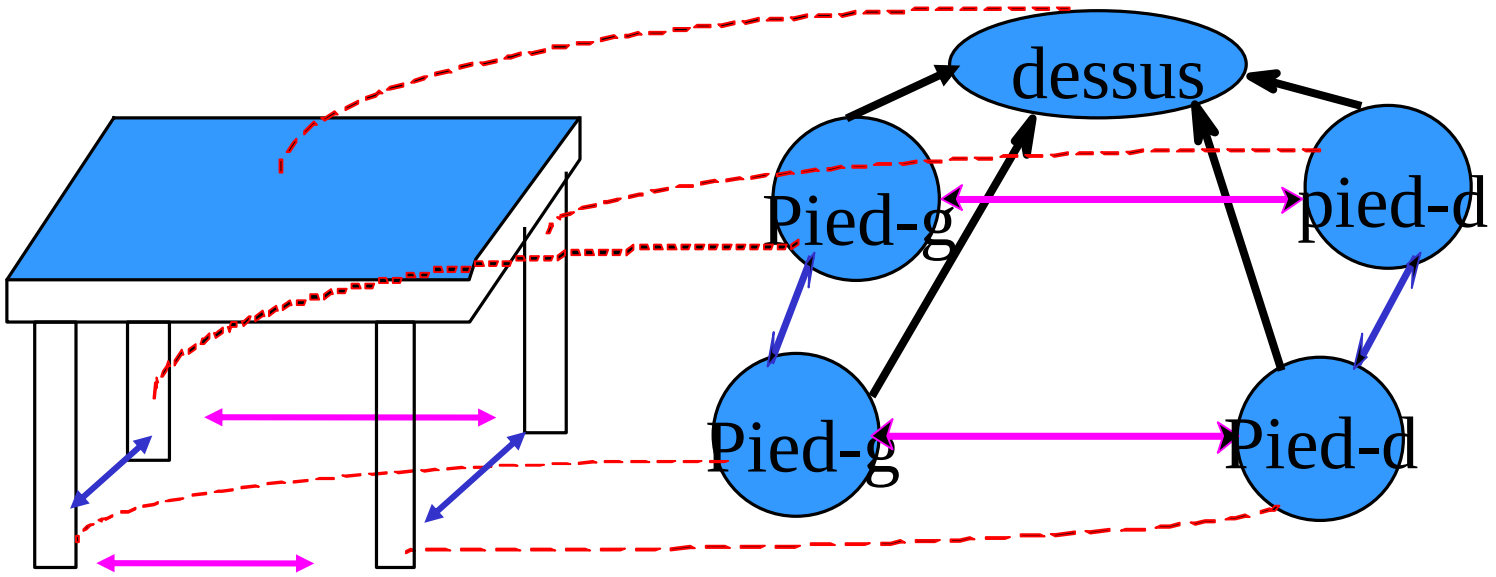


- Existence de plusieurs couches avec connections horizontales intracouches

- Relations top-down et bottom-up entre les couches

Relaxation discrète et Interprétation d'images

- associer chaque partie de l'image à un noeud d'un graphe sémantique



Contraintes spatiales :

sous **Arrière-avant** **Gauche-droite**

Relaxation discrète

- Appariement de graphes:
- Algorithme combinatoire.
- Etudier pour chaque couple de noeud la validité de la correspondance entre eux.
- Une correspondance est valide si les noeuds associés ont le même nombre d'arcs entrants et sortants

Relaxation discrète

- un outils de mise en correspondance entre une représentation et une forme: la relaxation discrète
- Les noeuds du graphe sont les primitives du modèle
- A chaque noeud est associé un ensemble d'étiquettes (régions segmentées) qui peuvent leur correspondre.
- Les arcs expriment les relations spatiales entre les primitives
- Principe: Eliminer récursivement toute interprétation non compatible avec les interprétations des noeuds environnants

Relaxation discrète

- Problème: les algorithmes de mise en correspondance peuvent être exponentiels

- Certaines combinaisons peuvent être éliminées en utilisant des contraintes naturelles

 La relaxation discrète utilise les graphes de cette façon.

Historique

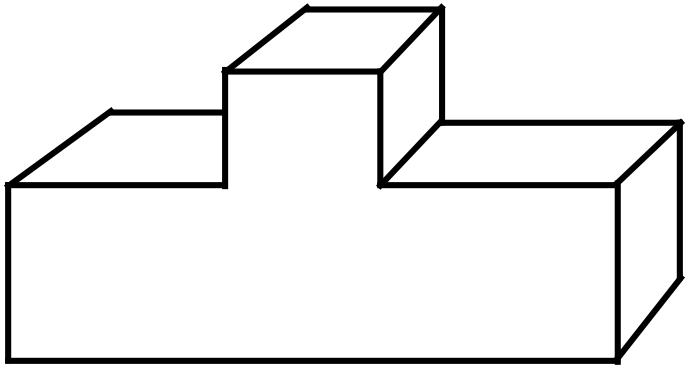
- Waltz (1975): scene labeling in the bloc world
- Rosenfeld, Zucker and Hummel (1976): scene labeling with relaxation operation
- Mohr and Henderson (1986): fast algorithm of arc-consistence checking (AC4)

Historique

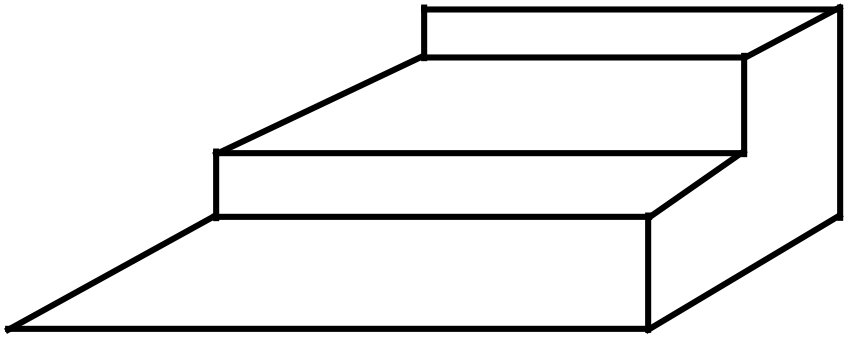
- Van Henteryck, Y. Deville, C.M. Teng (1992): AC5 algorithm.
- C. Bessière (1994): AC6 algorithm.
- A. Deruyver and Y. Hodé (1997): AC4bc algorithm, arc-consistence checking with two levels of constraint

Algorithme de waltz

- Permet de reconnaître des polyhèdres triédraux

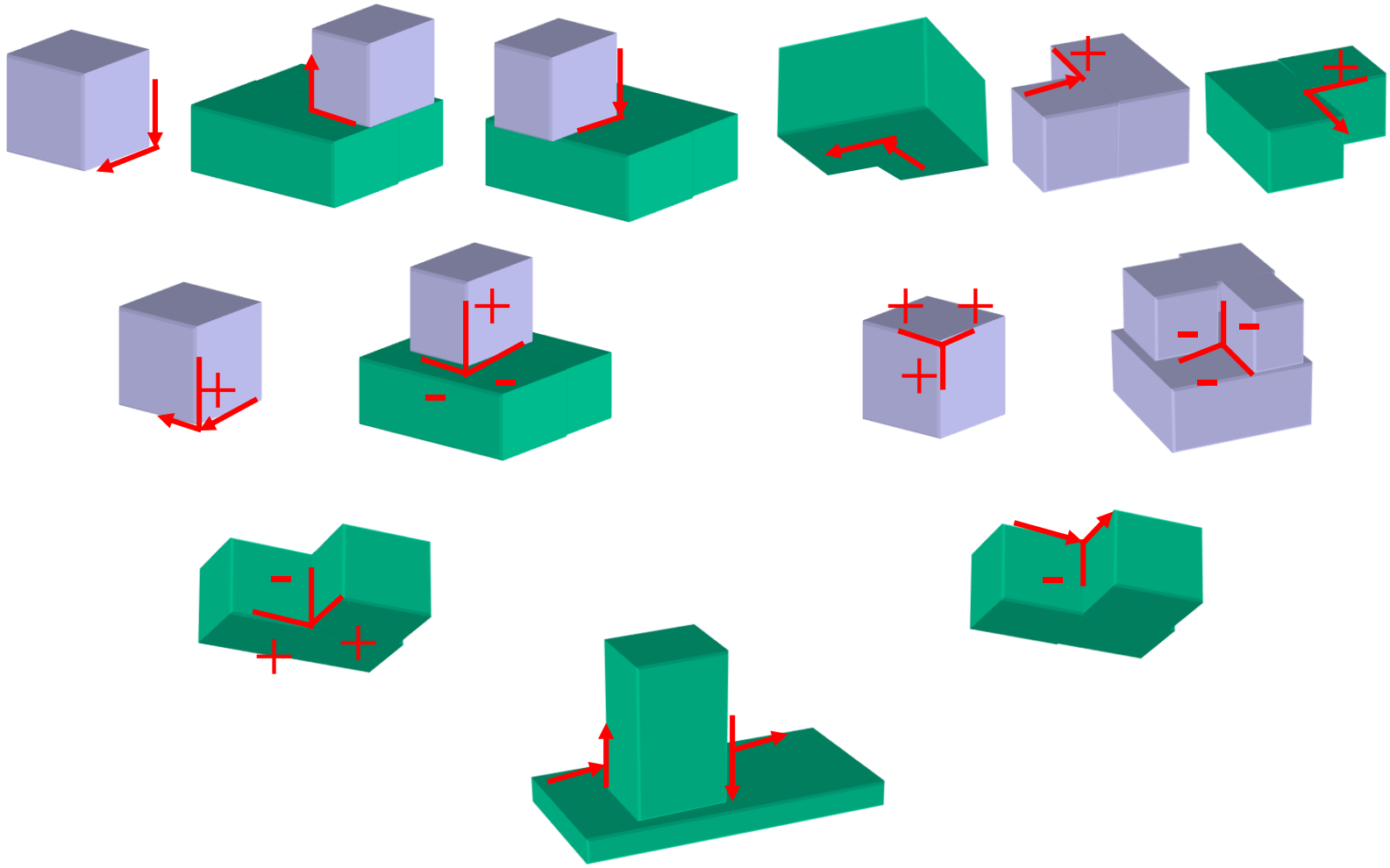


Objet possible

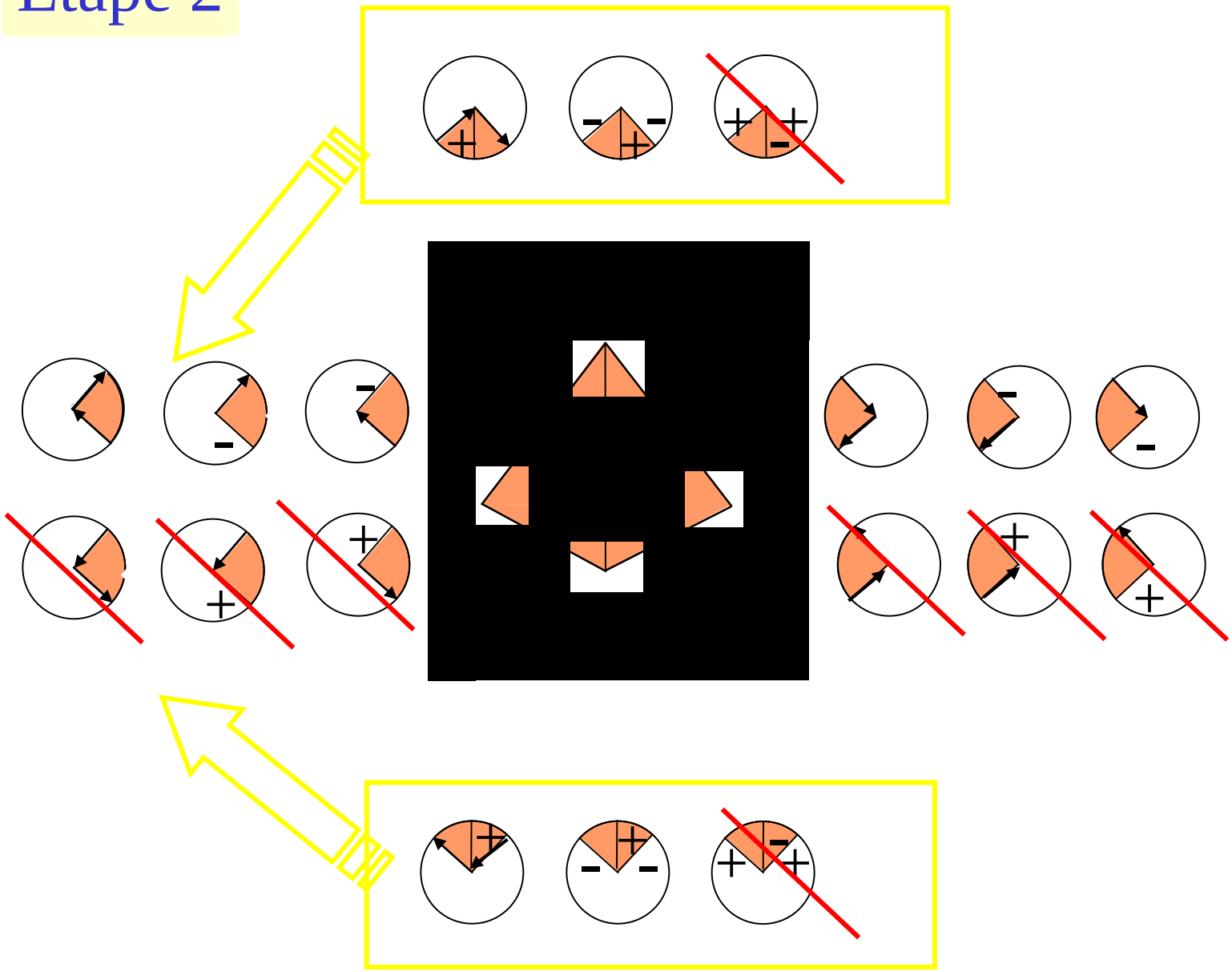


Objet impossible

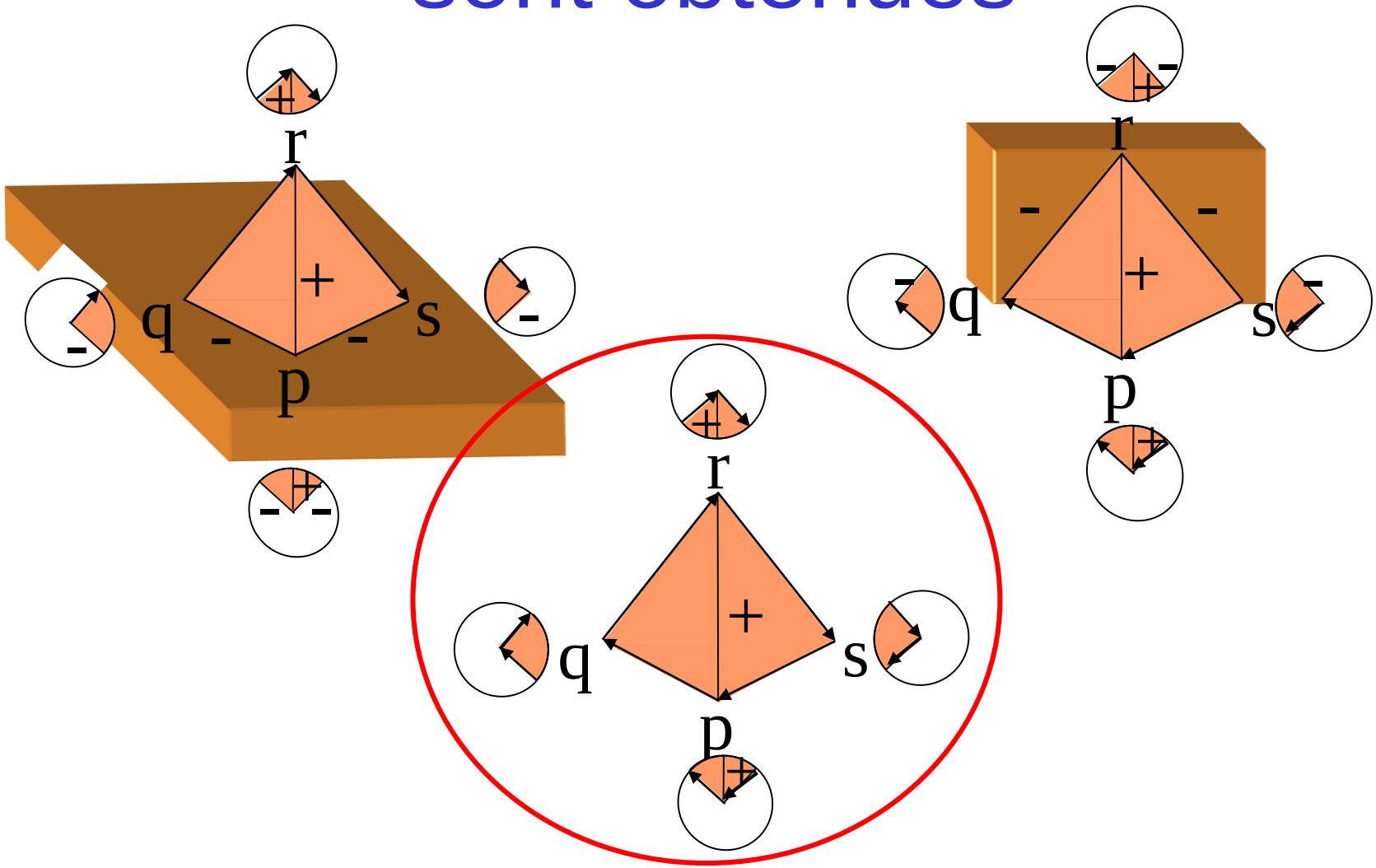
16 différents types de sommet



Etape 2



3 possibilités d'interprétation sont obtenues



Algorithme AC4

- Mohr et Henderson (1986)

Le contrôle de la consistence est un problème NP-difficile (très long en pratique)

mais

Beaucoup de problèmes réels peuvent être résolus en vérifiant la consistance d'arc.

Idée de base (1)- notations

- Soit un ensemble de variables représentée par des entiers 1, ..., n.
- Soit D_i le domaine associé à la variable i.
- Soit C_{ij} une contrainte reliant deux variables i et j, $C_{ij}(v,w)$ est vrai si $v \in D_i$ et $w \in D_j$ et (v,w) satisfait la contrainte C_{ij}.

Idée de base (2) - notations

- un graphe G est associé à un problème de satisfaction de contraintes si:
  - Un noeud “i” est associé avec chaque variable “i”.
  - Deux arcs orientés (i,j) et (j,i) sont associés avec une contrainte C_{ij}.
- arc(G) est l’ensemble des arcs de G, |arc(G)|=e
- node(G) est l’ensemble des noeuds de G, |node (G)|=n

idée de base (3)

- Definition 1: soit $(i,j) \in arc(G)$.

(i,j) est arc-consistant étant donnés $P(D_i)$ et $P(D_j)$ ssi $(\forall\ v \in D_i\ \exists\ w \in D_j\ ,\ C_{ij}(v,w))$.

- Definition 2: soit $P=P(D_1)\ x\ ...\ x\ P(D_n)$.

G est arc-consistant étant donné P ssi $\forall\ (i,j) \in arc(G)$, (i,j) est arc-consistant étant donnés $P(D_i)$ et $P(D_j)$.

- BUT: calculer le plus grand domaine arc-consitant pour G dans P.

algorithme

- InitQueue(Q)
- Pour chaque (i,j) ∈arc(G) faire
  - Pour chaque b ∈ Di faire
    - Total:=0
    - Pour chaque c ∈ Dj faire
      - Si Cij(b,c) alors
        - » Total := total+1
        - » E[j,c]:=E[j,c] ∪ {(i,b)}
      - Sinon rien
      - fsi
      - Si total =0 alors EnQueue(i,b,Q)
        - » Di:=Di-{b}
      - Sinon compteur[(i,j),b]:=total
      - fsi
    - fpour
  - fpour
- fpour

Elagage

Tantque non videQueue(Q) répéter

dequeue(j,c,Q)

pour chaque (i,b) ∈ E[j,c] faire

compteur[(i,j),b]:=compteur[(i,j),b]-1

si compteur[(i,j),b]=0 alors

enqueue(i,b,Q)

Di:=Di-{b}

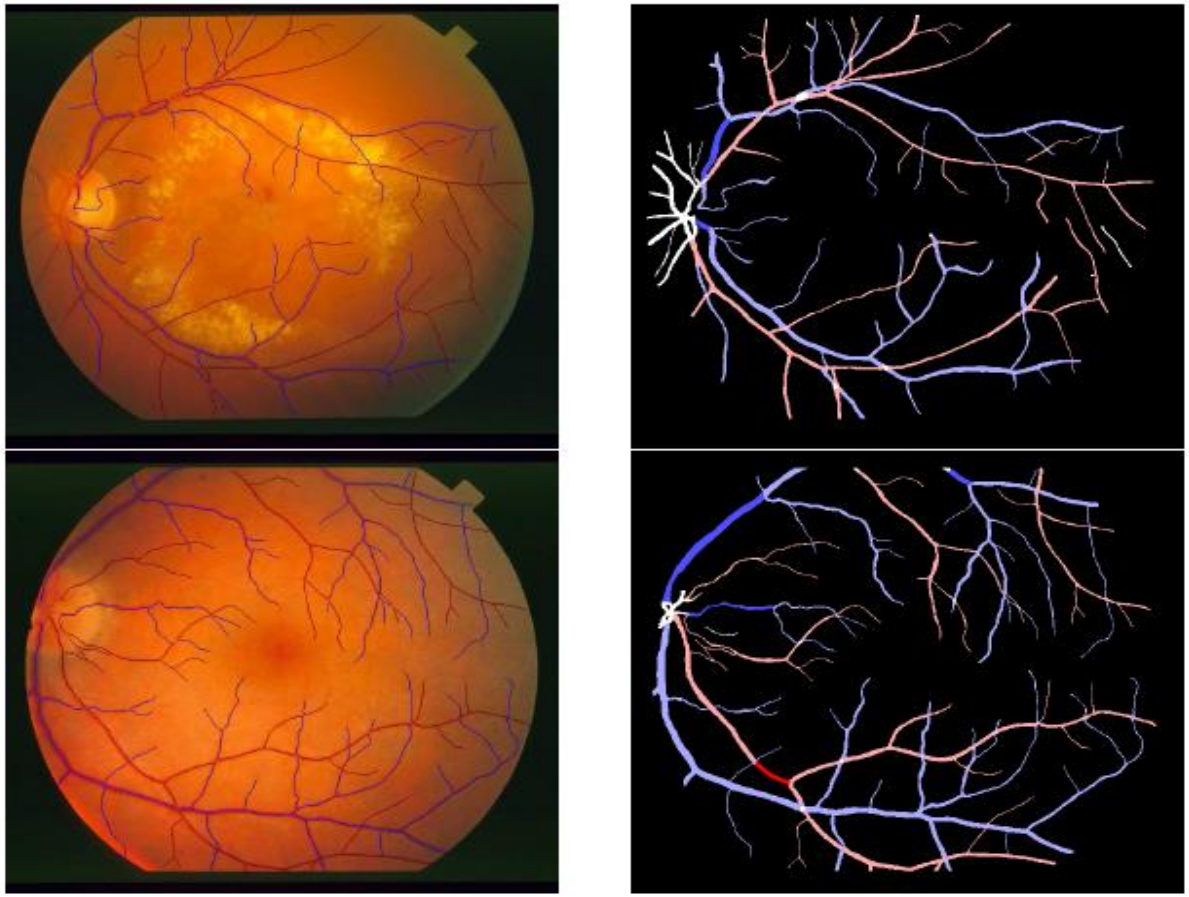
sinon rien

fsi

fpour

ftq

Exemple: étiquetage du graphe vasculaire rétinien (artères et vaisseaux)

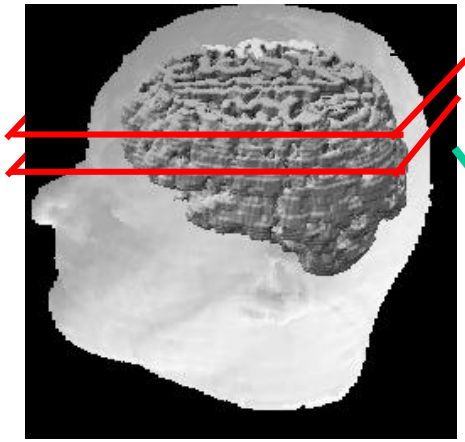


Limites

- On suppose qu'à un noeud est associé une seule valeur.

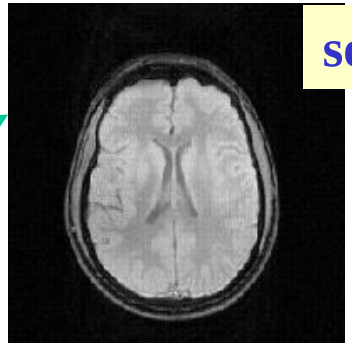
- Il est supposé que la mise en correspondance est bijective.

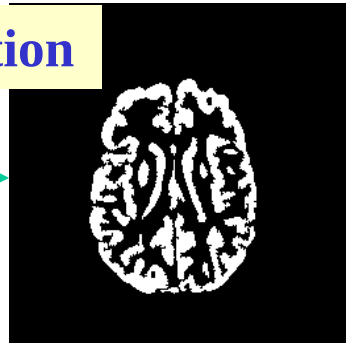
Très rare dans la pratique

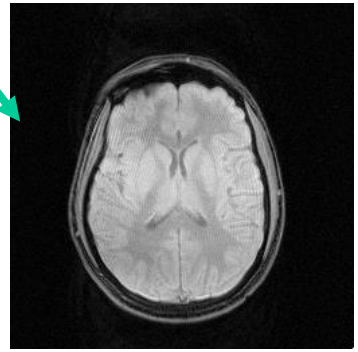


IRM 3D

segmentation



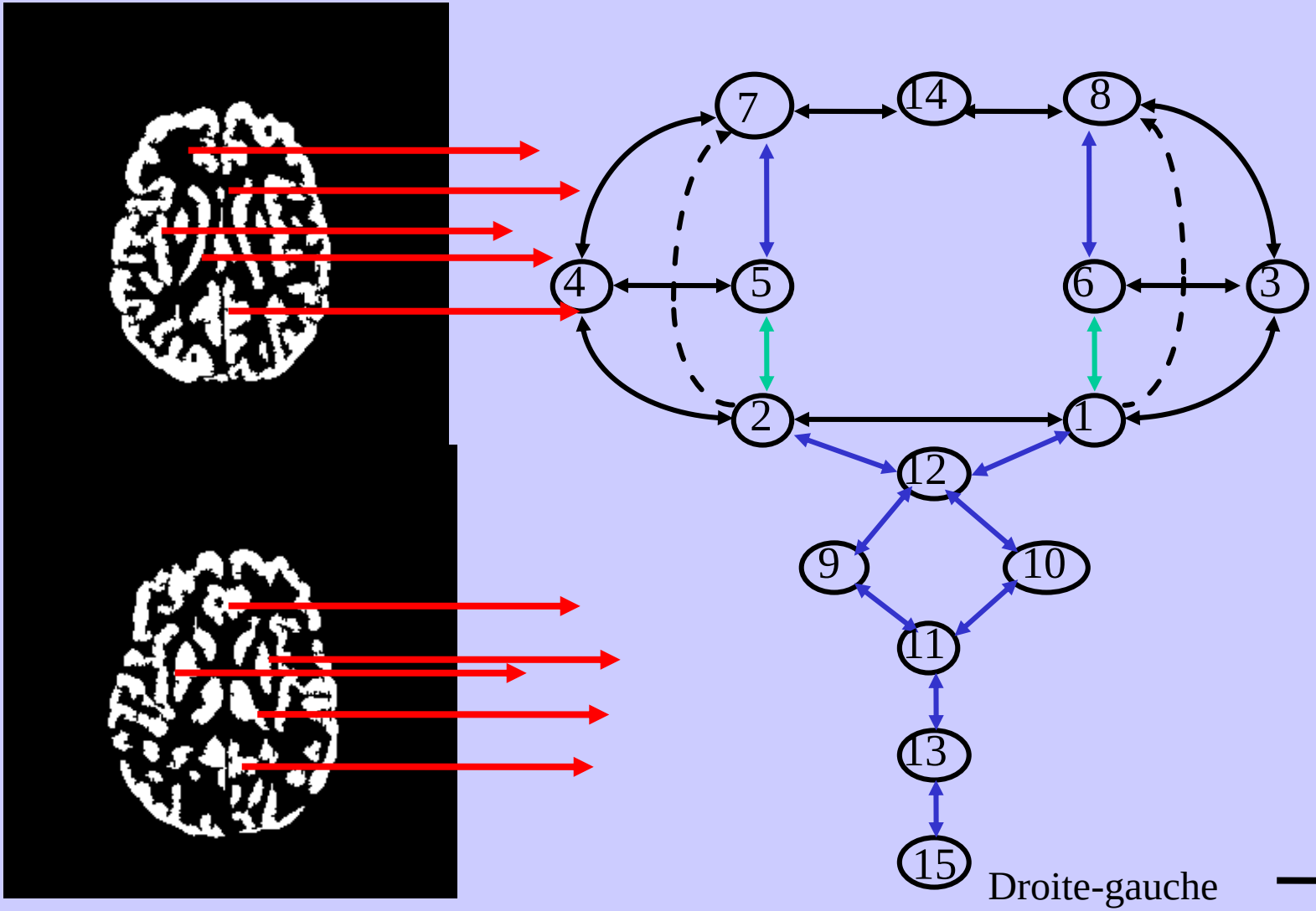




coupes

images en niveaux de gris

images segmentées


7
14
8
4
5
6
3
2
1
12
9
10
11
13
15
Droite-gauche
Dessus-dessous
Devant-derrière
Dessous et en arrière

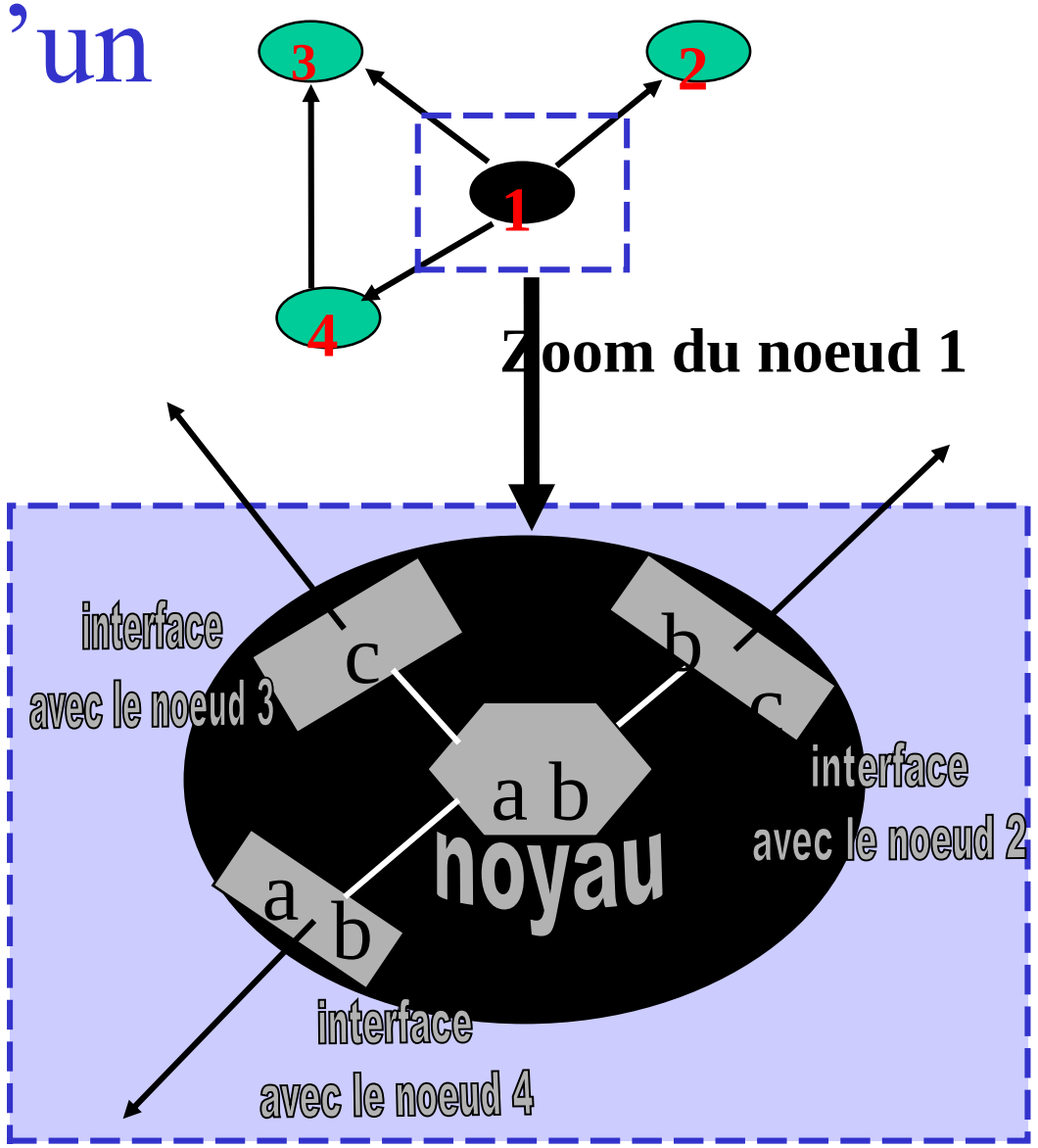
Solution: satisfaction de l'Arc-consistance à deux niveaux de contraintes (Artificial Intelligence, 1997)

- Introduction de la notion de contraintes "intra-noeud".

- Introduction d'une nouvelle structure de noeud

Structure d'un noeud

- Un noeud est constitué :
  - D'un noyau K_i
  - D'un ensemble d'interfaces X_i

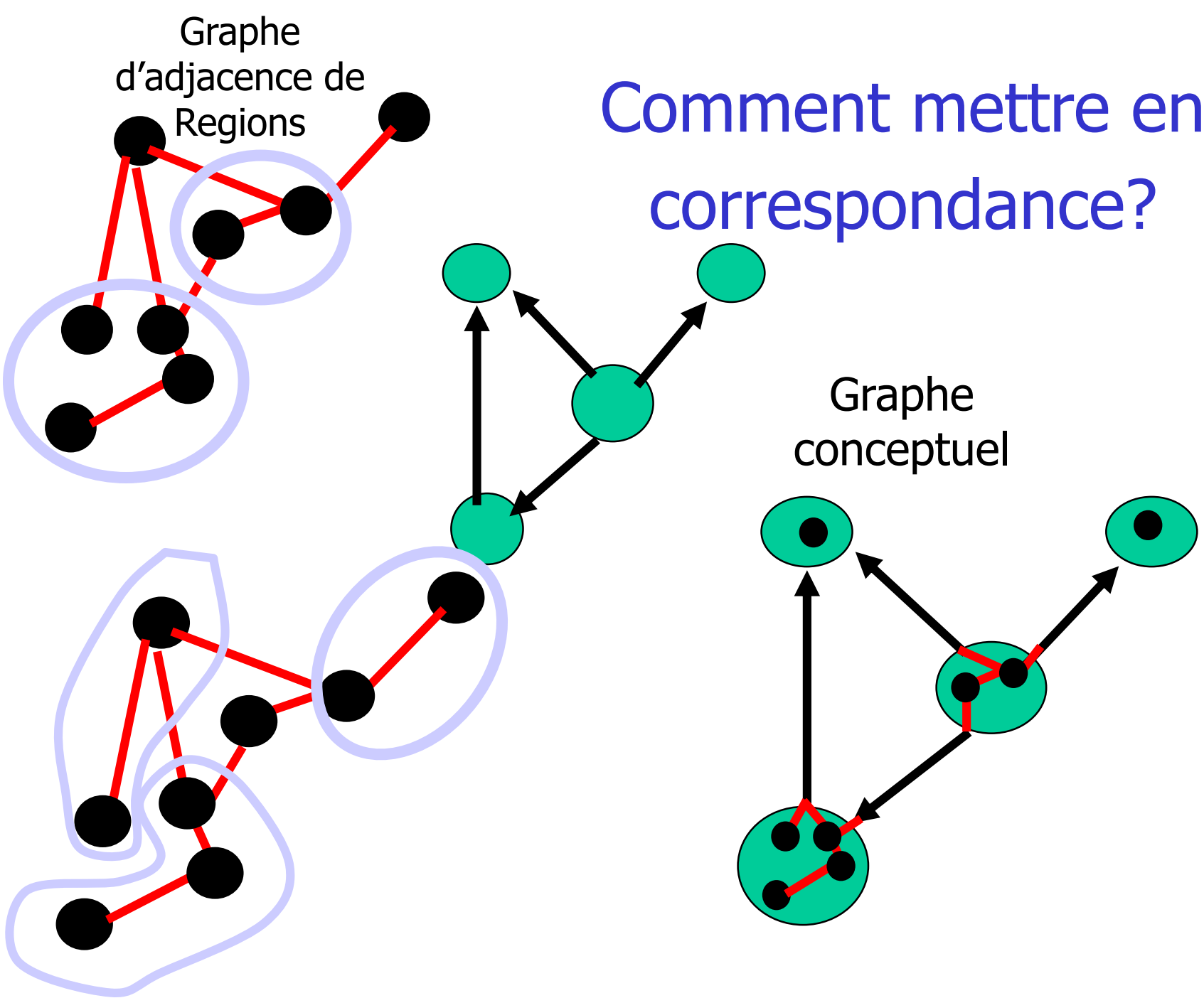


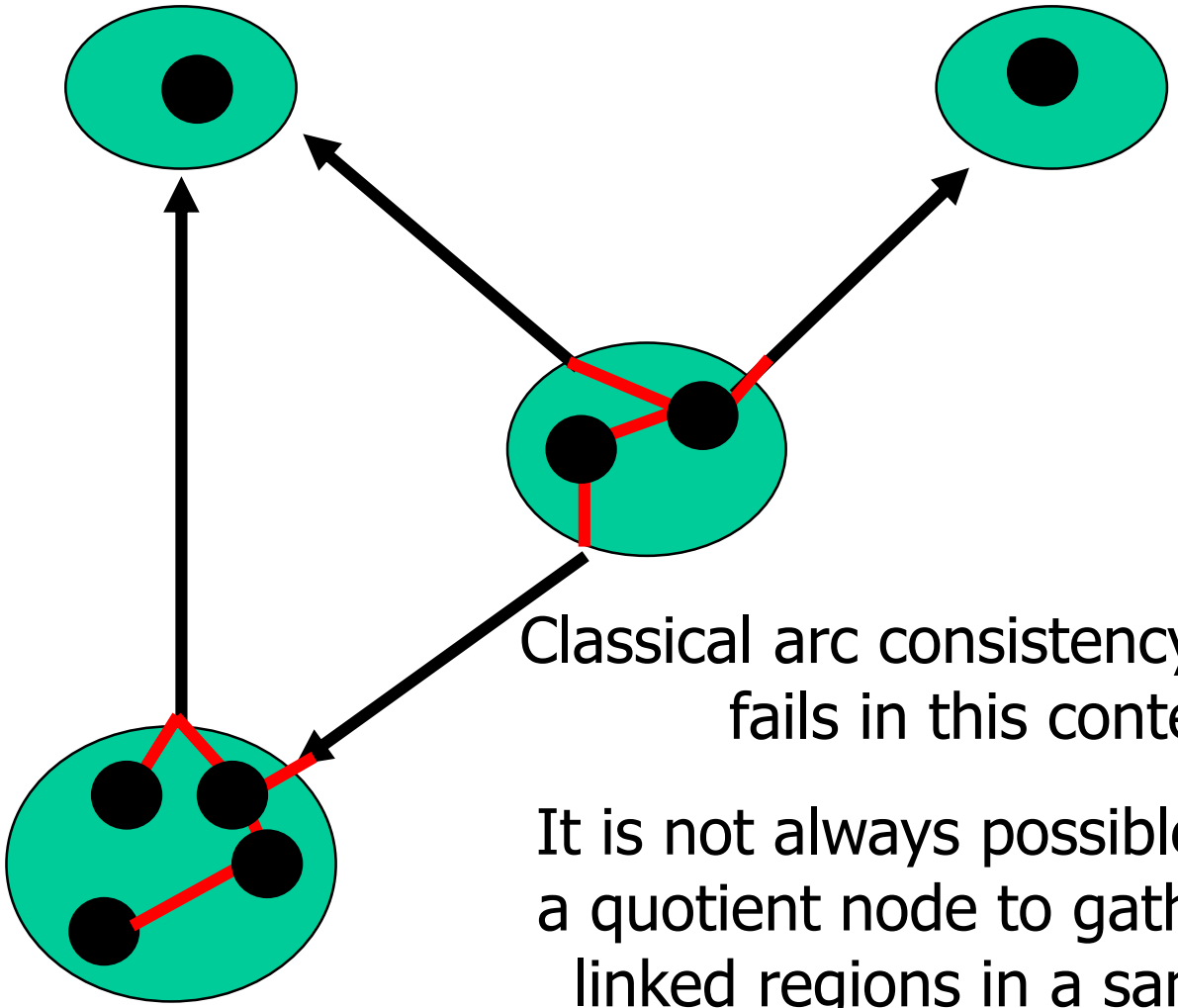
Problème de satisfaction de contraintes à deux niveaux de contrainte

- Definition:
- Let Cmpi be a compatibility relation such that (a,b) $\in$ Cmpi $\Leftrightarrow$ a and b are compatibles.
- Let Cmpgi be a global compatibility constraint applied on a set of values Si $\subset$ Di such that Si $\in$ Cmpgi $\Leftrightarrow$ Si satisfies the global compatibility constraint.
- Let Cij be a constraint between i and j.
- Let Si, Sj such that Si $\subset$ Di and Sj $\subset$ Dj, Si, Sj $\mapsto$ Cij means that (Si, Sj) satisfies the oriented constraint Cij.
- Si, Sj $\mapsto$ Cij $\Leftrightarrow$ $\forall$ai $\in$ Si, $\exists$(a'i, aj) $\in$ Si x Sj, such that (ai, a'i) $\in$ Cmpi and (a'i,aj) $\in$Cij and $\forall$aj $\in$Sj, $\exists$(a'j,ai) $\in$ Sj x Si, such that (aj, a'j) $\in$ Cmpj and (ai,a'j) $\in$ Cij.
- The sets {S1 ... Sn} satisfy $PSCDF_{DNC}$

  $\Leftrightarrow$ $\forall$ Cij Si, Sj $\mapsto$ Cij et Si $\in$ Cmpgi and Sj $\in$ Cmpgj.

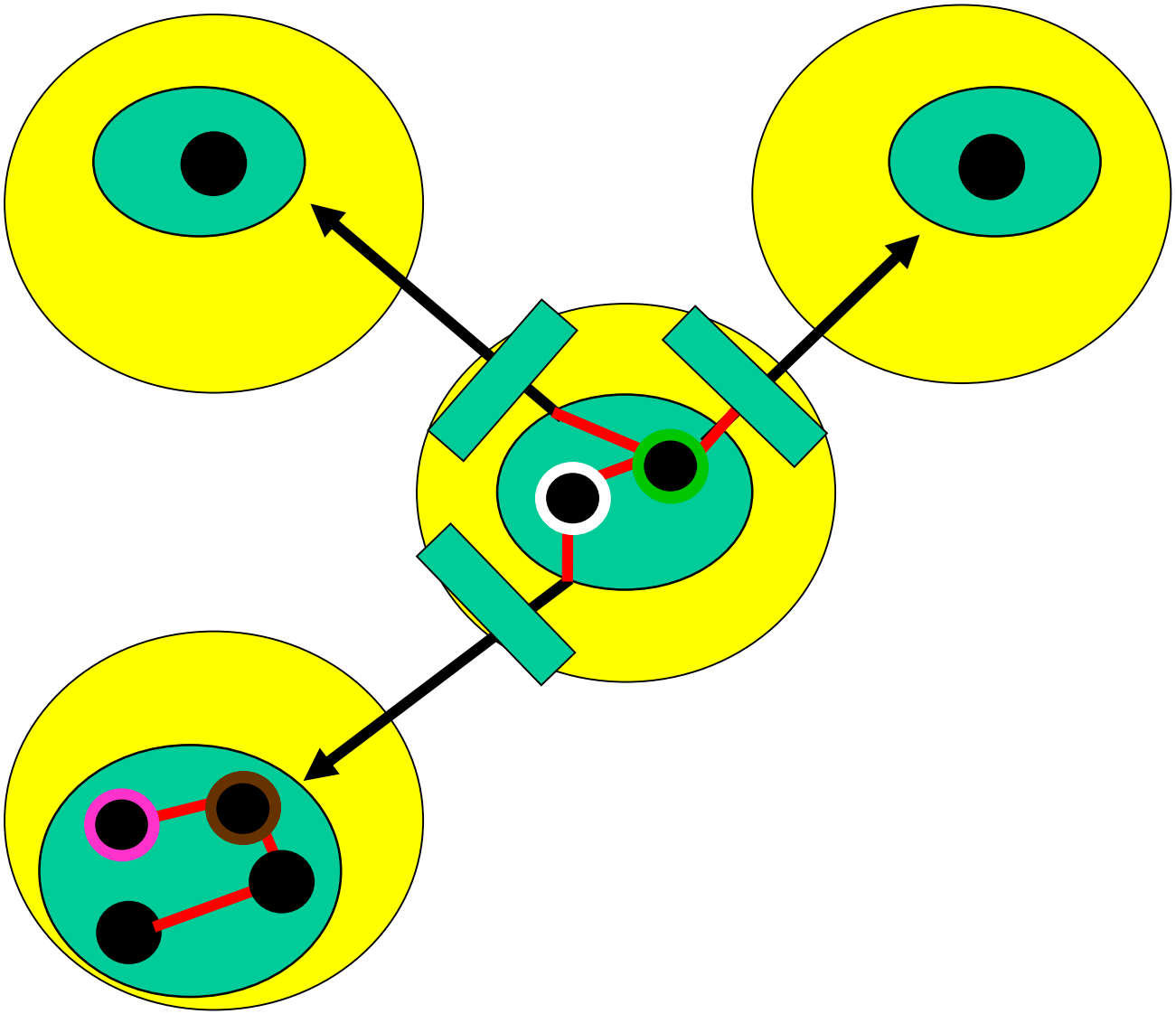
Comment mettre en correspondance?

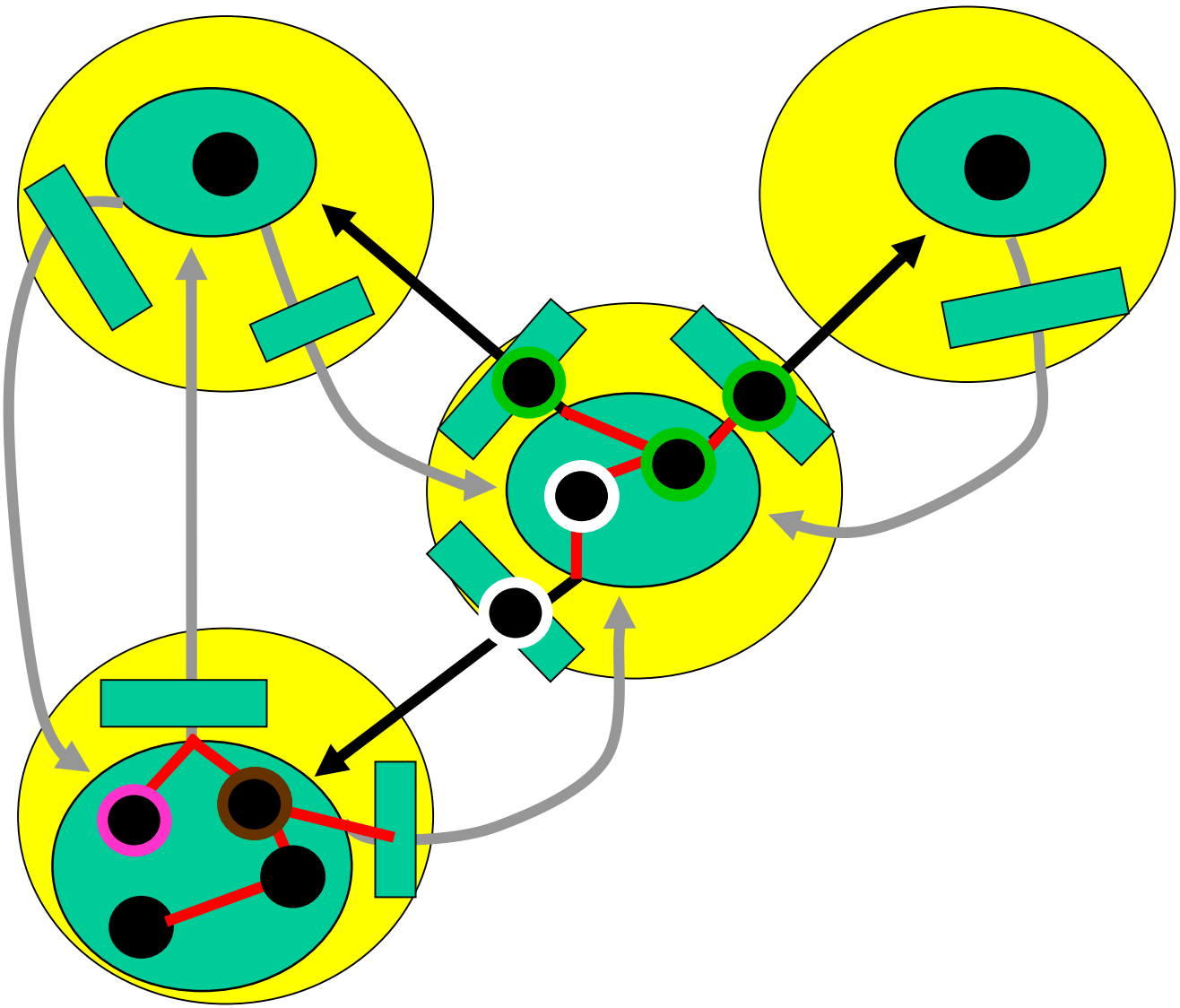




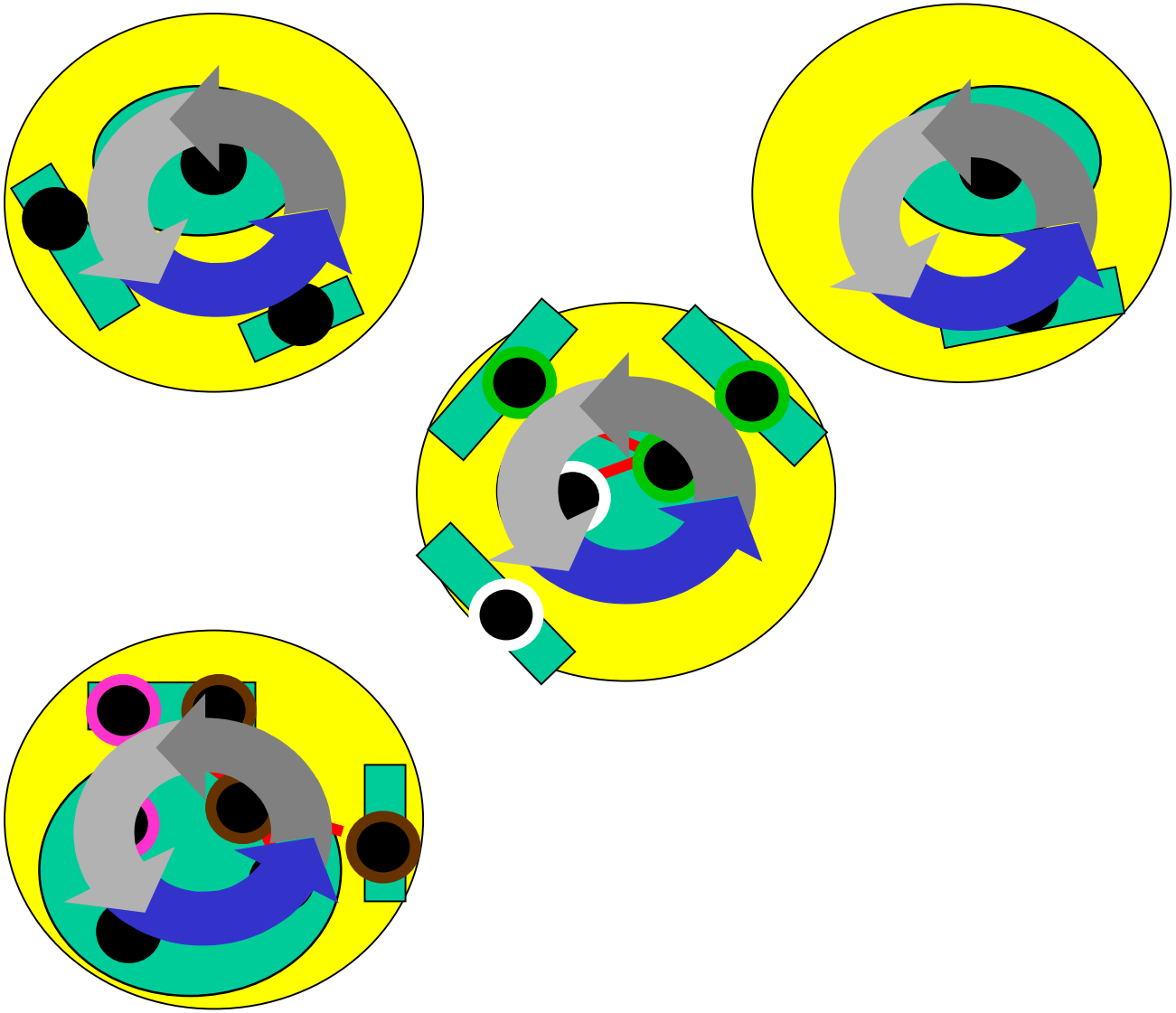
Classical arc consistency checking fails in this context

It is not always possible de build a quotient node to gather all the linked regions in a same node

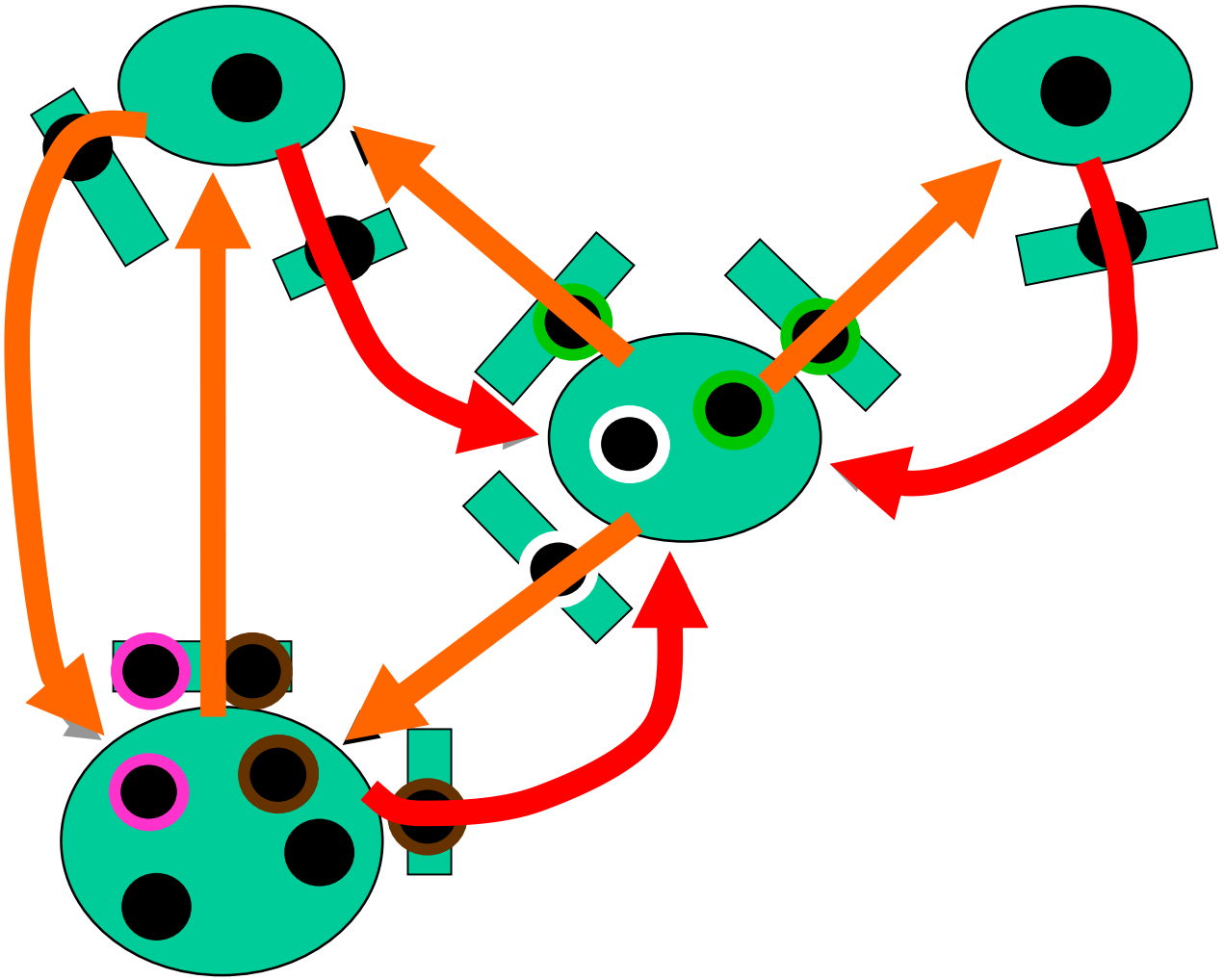




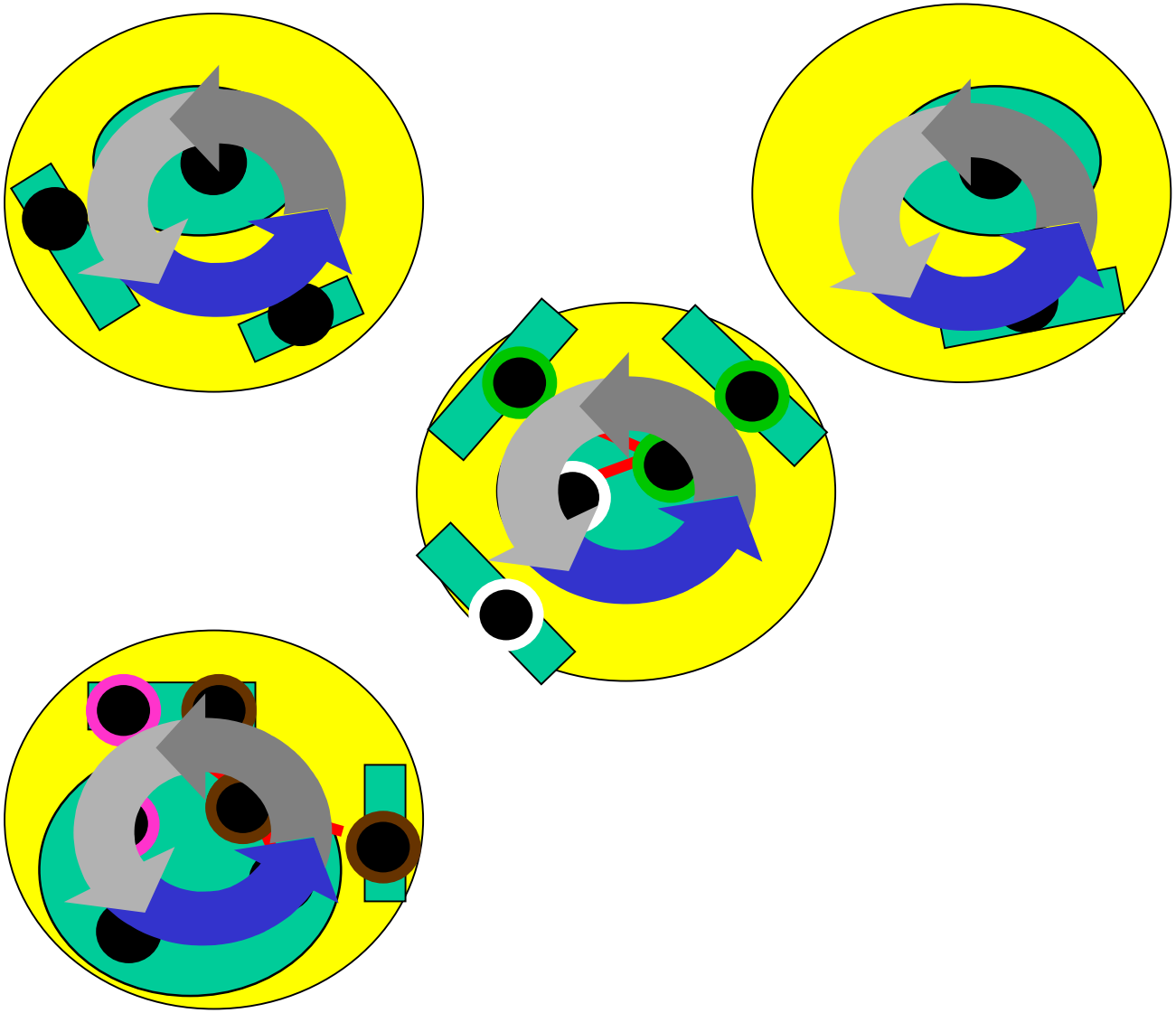
2nd niveau : contraintes intra-noeud

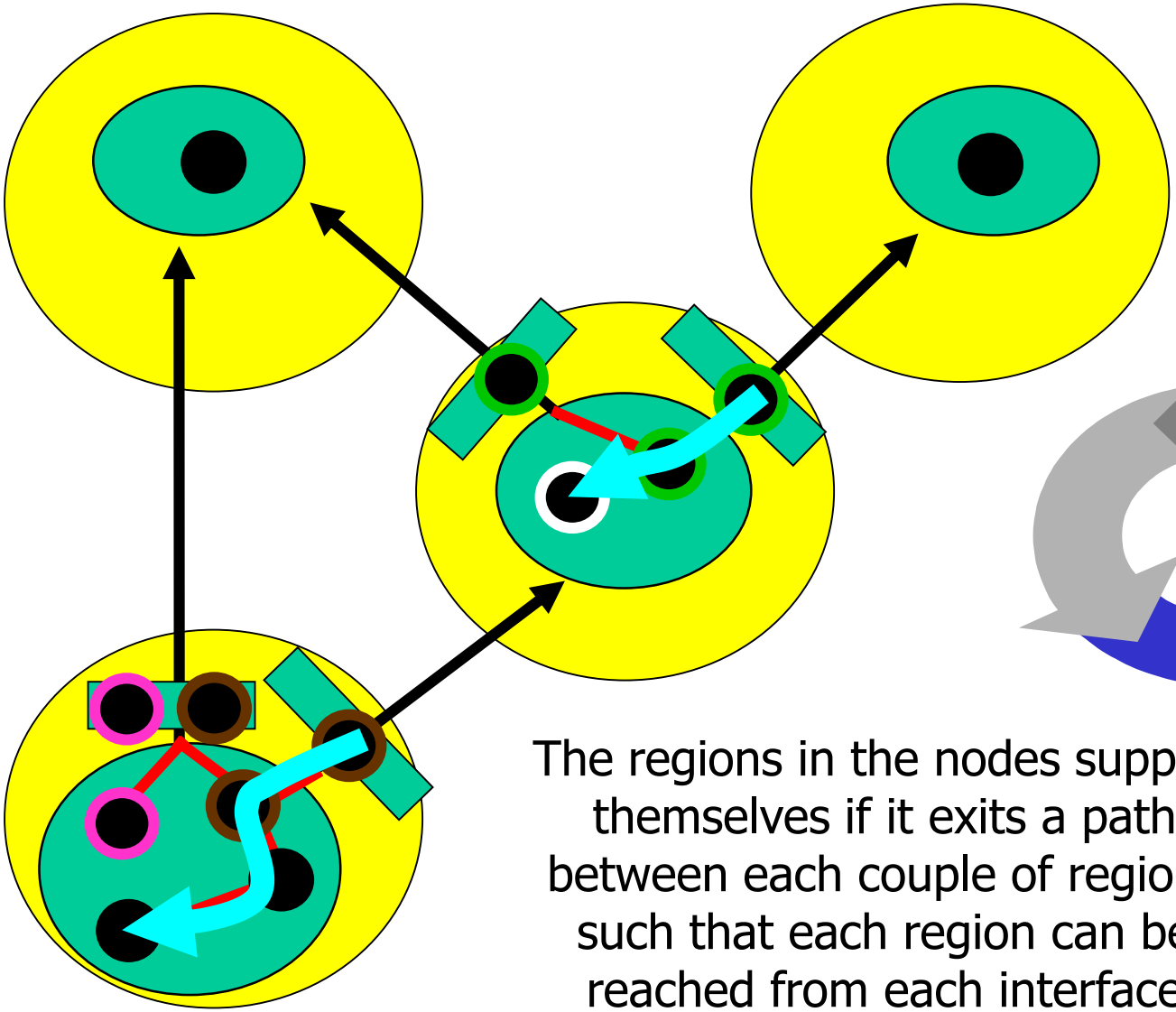


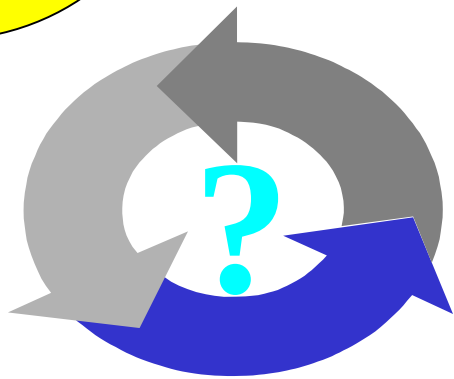
Premier niveau : contraintes inter-noeud



2nd niveau : contraintes intra-noeud



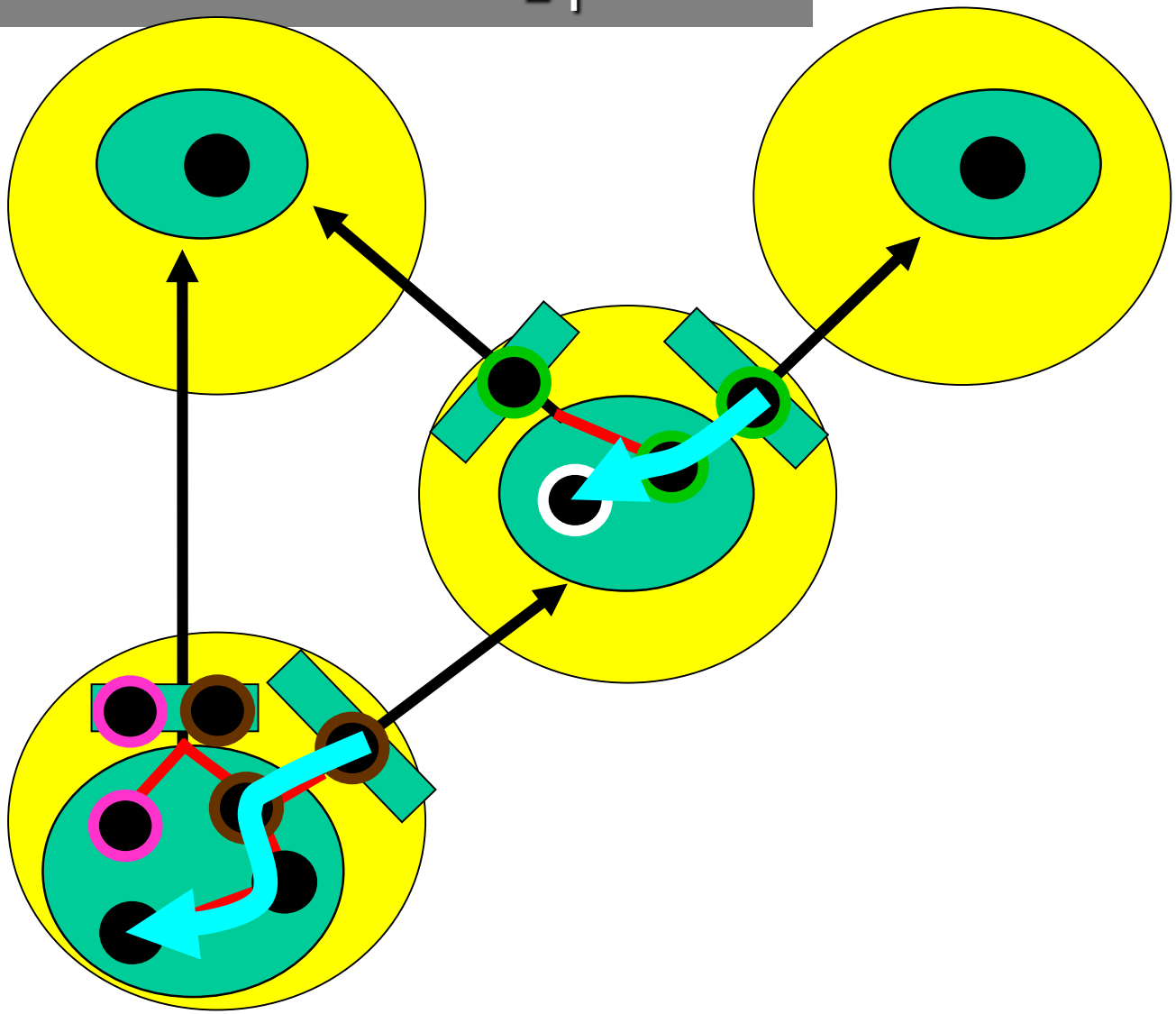




The regions in the nodes support themselves if it exits a path between each couple of regions such that each region can be reached from each interface

A path = succession of spatial relations

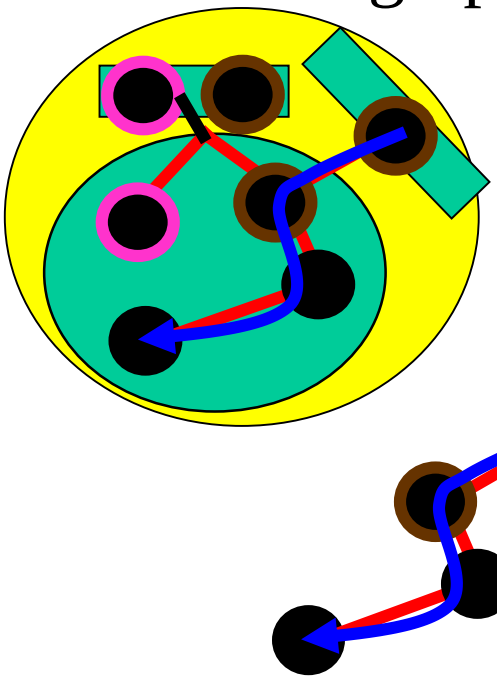
defined in a finite set Σ_T

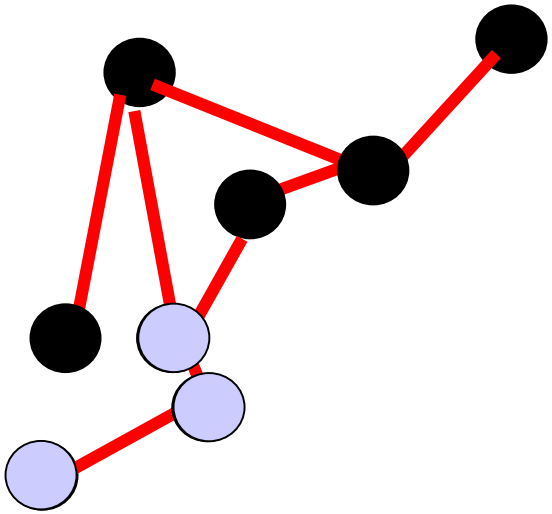


Intra-node constraints: compatibility between two values (C_{mpi})

Finding a succession of values (regions) able to link two values that can be associated with a node of the semantic graph.

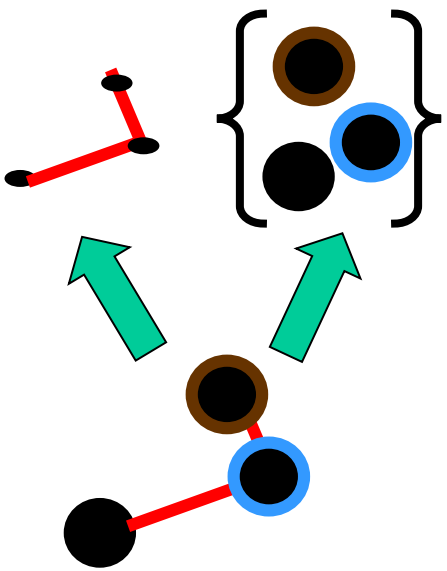
➔ Finding a linear sub-graph of the region adjacency graph.





Intra-node constraints: compatibility between two values (C_{mpi})

Finding a succession of values (regions) able to link two values that can be associated with a node of the semantic graph.



➔Finding a linear sub-graph of the region adjacency graph.

➔Two kinds of constraints on the graph:

- Path constraints applied on the arcs.
- Constraints applied on the set of nodes (union of regions associated with the nodes)

Propriétés

- Théorème 1: AC4bc se termine toujours
- Théorème 2: G est arc-consistant lorsque AC4bc se termine
- Théorème 4: la complexité de temps de AC4bc est en $O(en^3d^2)$ dans le pire des cas.

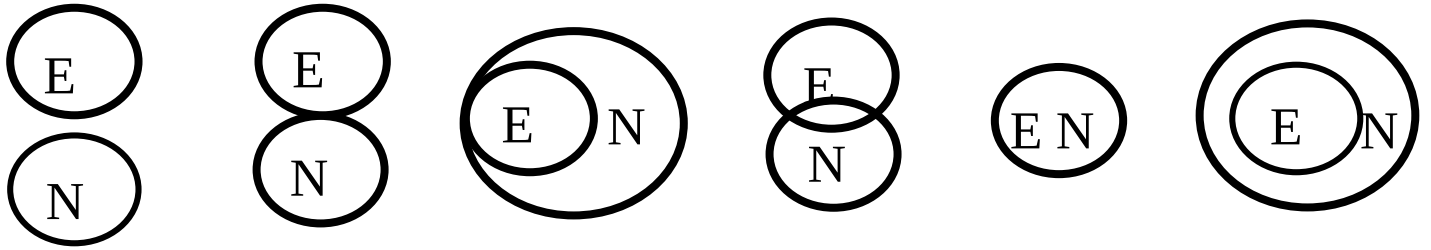
Interprétation d ’Image par relaxation discrète: Introduire des relations spatiales complexes

Complex spatial relations

- Our model: a conceptual graph describing very precisely the spatial organization of the different parts of the object that we look for.
- Adjacency relations are too poor
- Some systems of complex spatial relations have been proposed.

Complex spatial relations

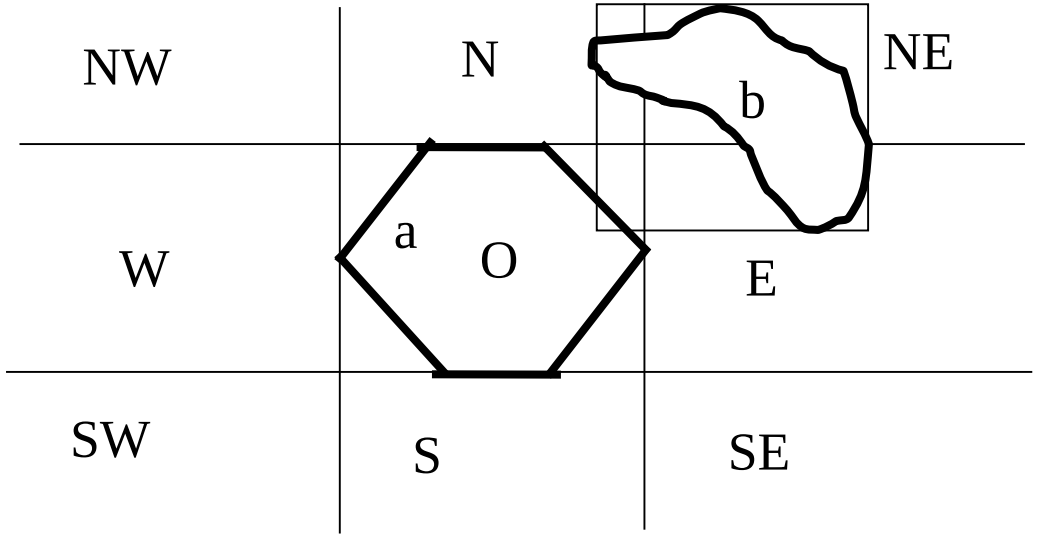
- The RCC8 system: it proposes 8 elementary relations in a topologic context:



- Does not take into account the shape of the regions and the directional information.

Complex spatial relations

- Cardinal Direction Relation Formalism (Skiadopoulos and Koubarakis)
- Use the notion of minimum bounding box (powerful reduction of the information)

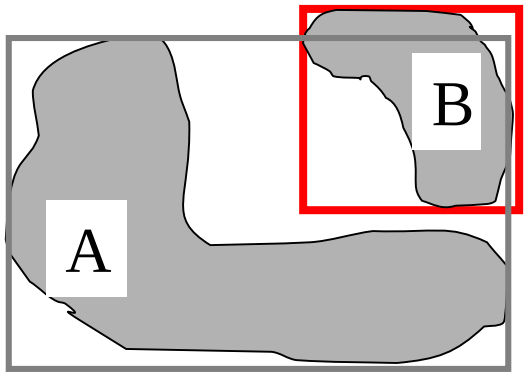


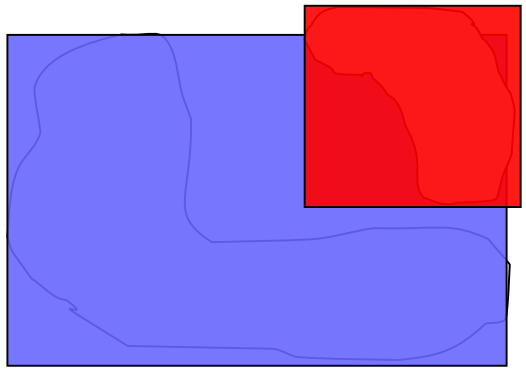
Complex spatial relations

- RCC8 and CDRF are powerful formalisms.
- It is possible to retrieve the RCC8 relations from the CRDF.

  - But there is no notion of distance

  - The minimum bounding boxes are not always enough to compute correct distances:

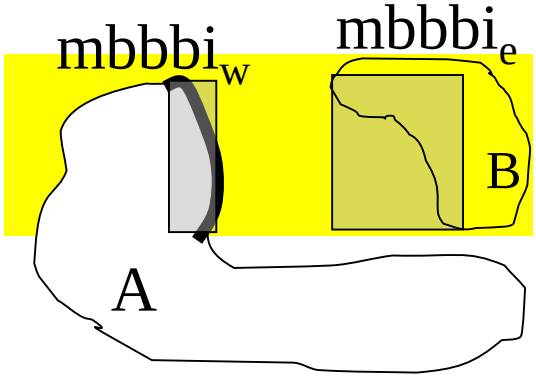




Connectivity-Direction-Metric Formalism

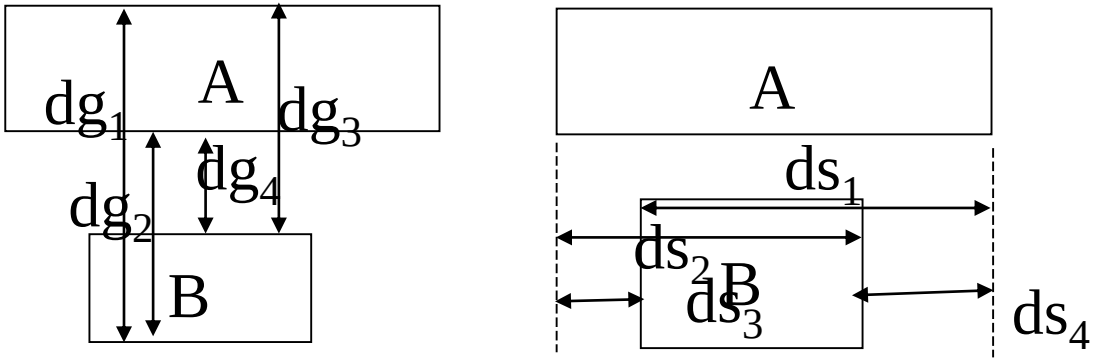
3 kinds of basic information:

- The connectivity C(x,y) “x is connected to y”
- The notion of minimum bounding box
- A new notion of minimum bounding box of border interface.



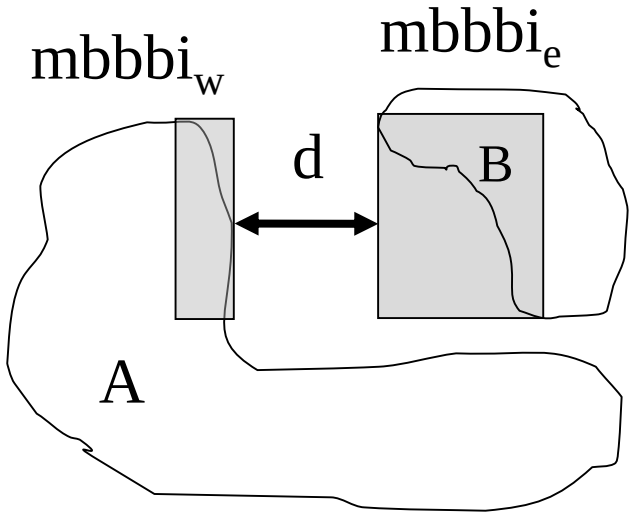
Connectivity-Direction-Metric Formalism

- 8 distances can be defined between minimum bounding boxes:
- Example:



Connectivity-Direction-Metric Formalism

- We can define the distance d between two minimum bounding boxes of border interface:



Connectivity-Direction-Metric Formalism

An elementary relation is a relation:

- (1) of connectivity or non connectivity
- (2) of directional relationships between mbb with none or one metric relation chosen among the metrics dsi and dgi (i=1…4) (with inferior and superior limits). We have 4 directional relations: N (North), S (South), W (West) et E (East).

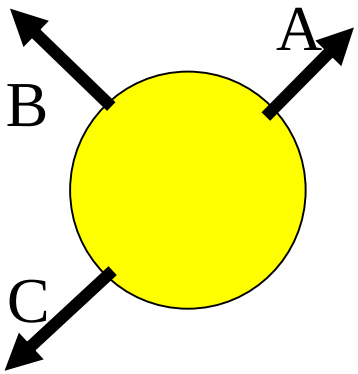
Allows to retrieve the CDRF

- (3) of directional relationships between mbbbi with one metric relation (with inferior and superior limits). We have 4 directional relations: Ni, Si, Wi et Ei.

How to combine these relations?

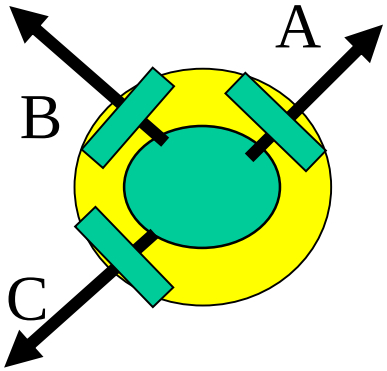
- Example: (Ei or Wi) and (Si or Ni)

(1)



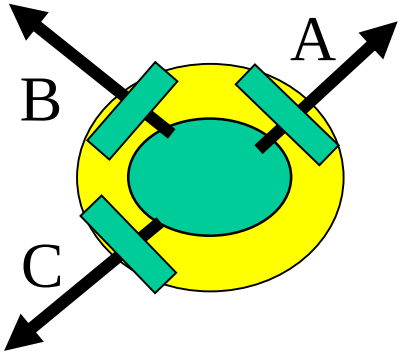
With classical arc-consistency checking
A and B and C must be satisfied for all the labels

(2)



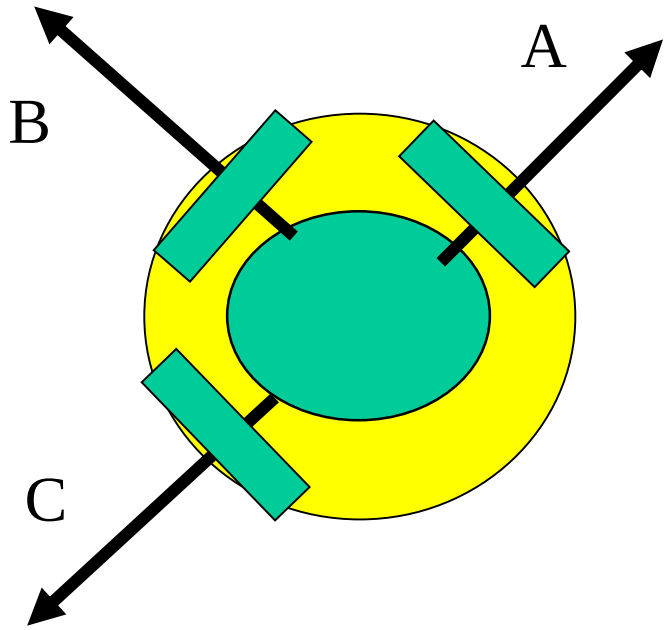
With arc-consistency with bilevel constraints
A and B and C must be satisfied but not necessary directly for all the labels

(3)



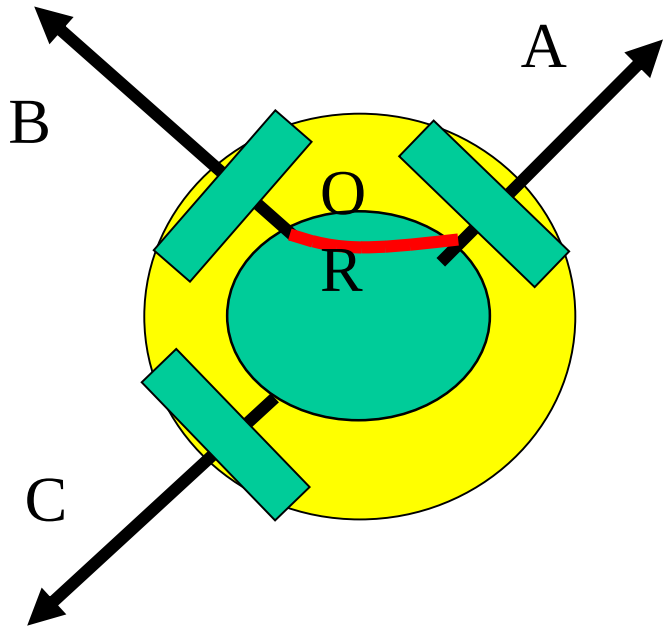
With quasi arc-consistency more complex constraints can be expressed :
With number of relaxation equal to
$\rightarrow 0$ A and B and C must be satisfied as for case (2)
$\rightarrow 1$ (A and B) or (A and C) or (B and C) must be satisfied
$\rightarrow 2$ A or B or C must be satisfied

We want more !



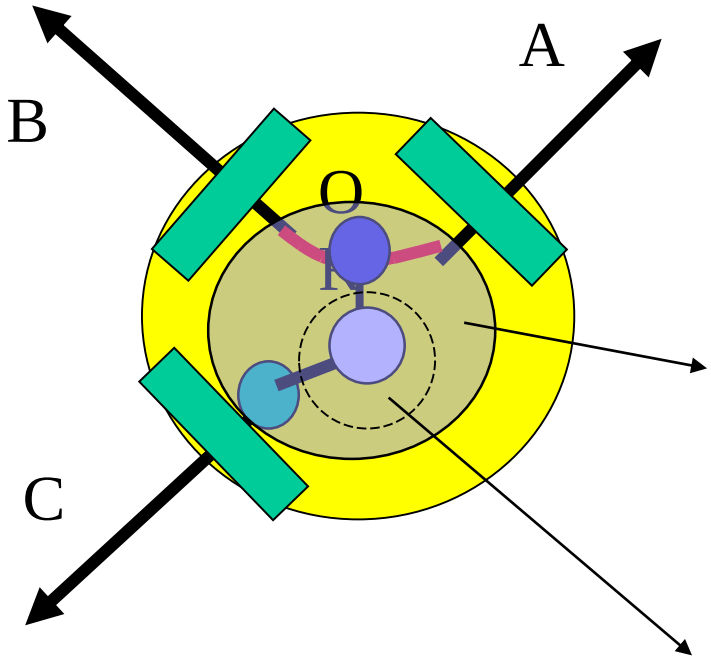
We want (A or B) and C

We want more !



We want (A or B) and C

We want more !

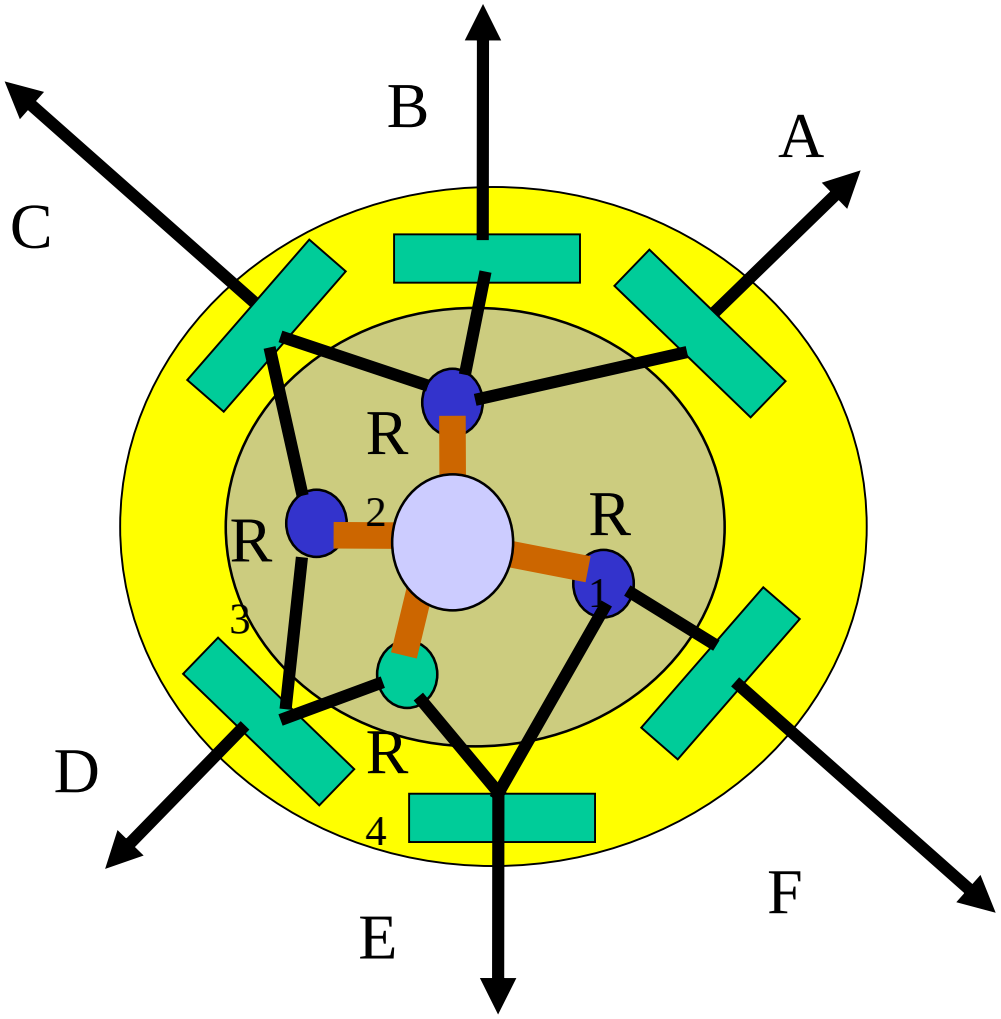


We want (A or B) and C

The kernel is divided in two new levels

•The shell : nodes linking interfaces according to logical combinations of (spatial) constraints (linked with a “OR” logical relationship)

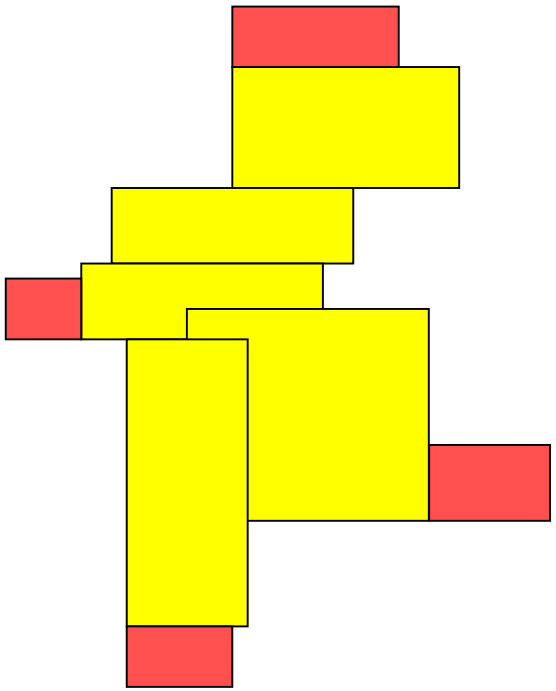
We can linked one interface with different nodes of the shell and described any logical combination



R_1 , R_2, R_3 , et R_4 are the number of authorized relaxations for each node of the shell.

Example of logical combination of spatial relations:
(A or B or C) and (C or D) and (D or E) and (E or F)

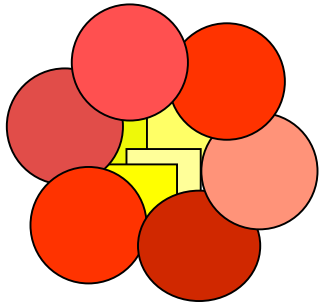
An example with the relation « is around of » to describe a flower:



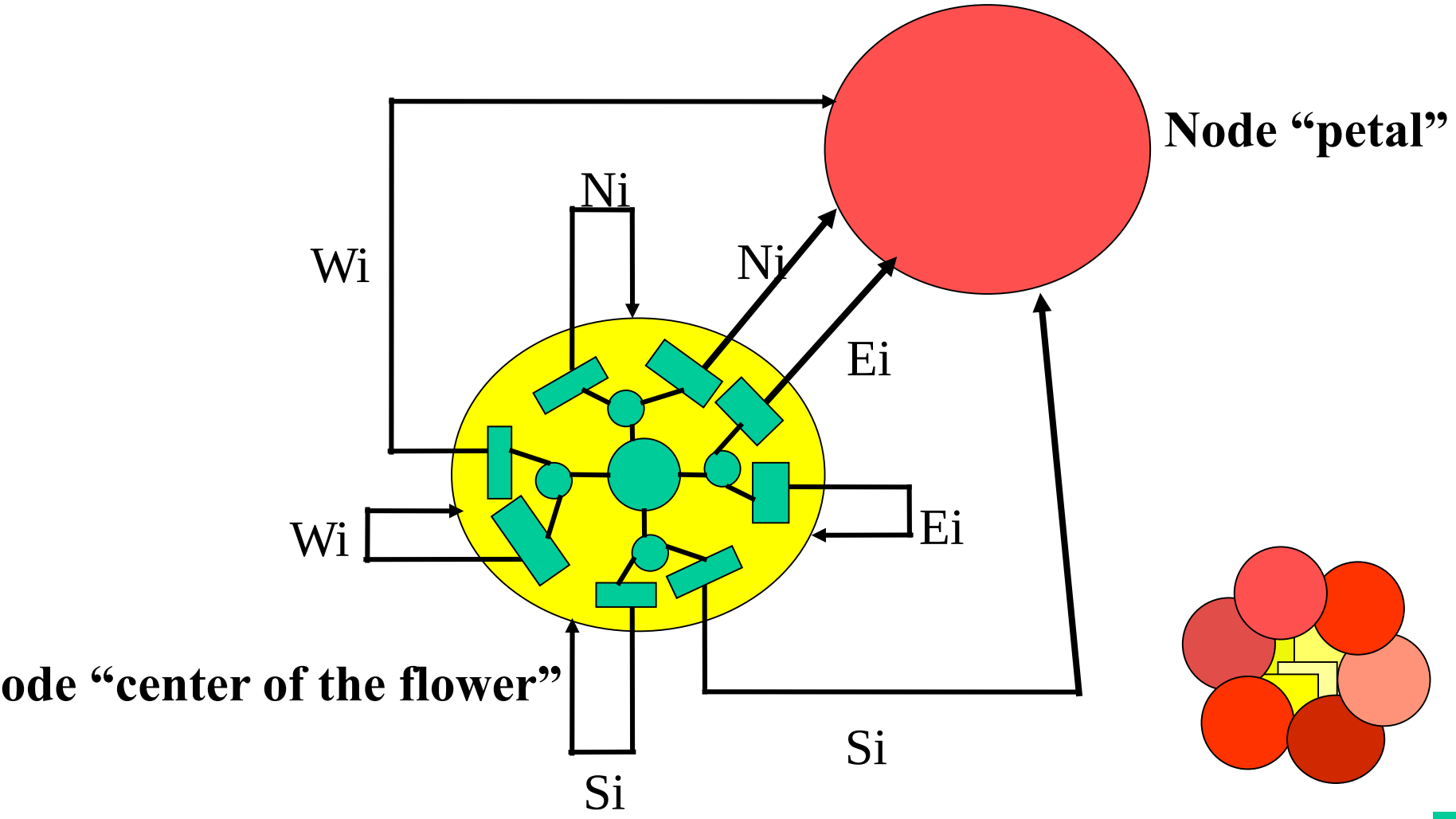
A region center of the flower is defined recursively by : a region center has

a South-neighbor which is a center or a petal
AND a West neighbor which is a center or a petal
AND a North neighbor which is a center or a petal
AND an East neighbor which is a center or a petal

•An intra-node constraint Cmpi is not a sufficient constraint (see left example)

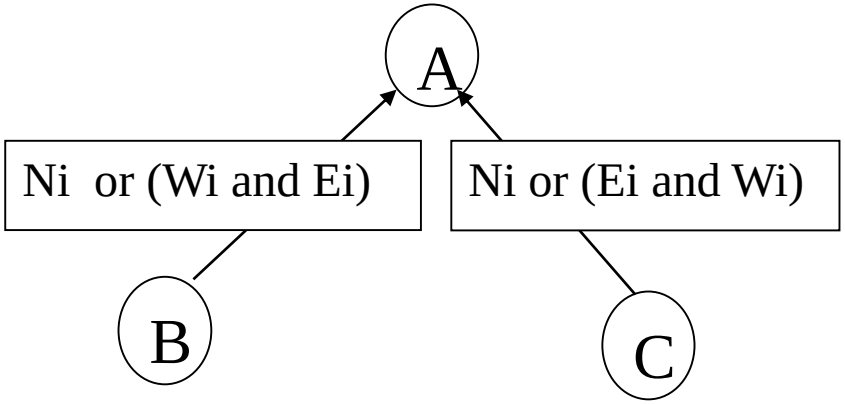


An example with the relation « is around of » to describe a flower:

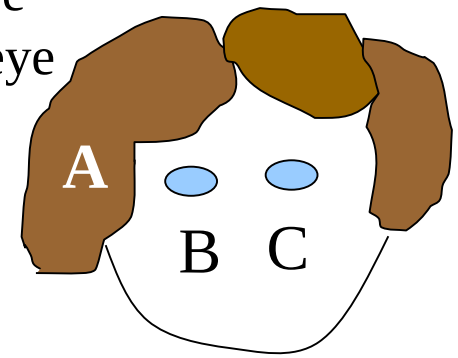


Connectivity-Direction-Metric Formalism

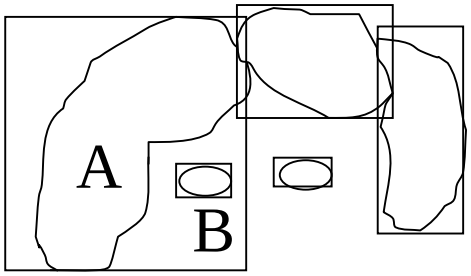
- An example with the relation “is partially around with a given distance »: the case of the hairs around the eyes

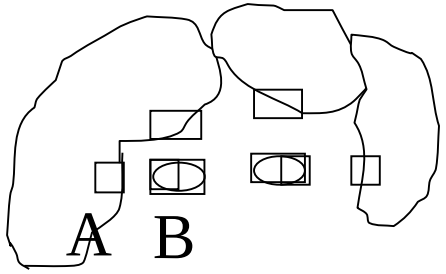


A : hairs
B :left eye
C: right eye



Connectivity-Direction-Metric Formalism





Classical minimum bounding boxes: overlapping = it is not possible to compute a distance between A and B.

Minimum bounding boxes of border interface: Better possibilities of computation of distances.

Experimentations

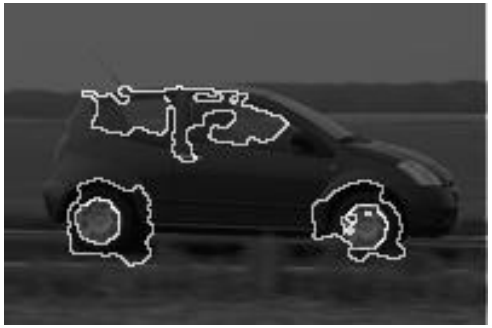
- On human faces:

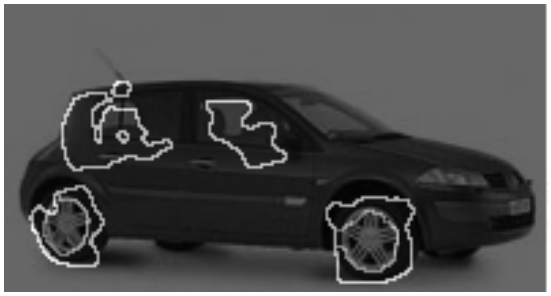




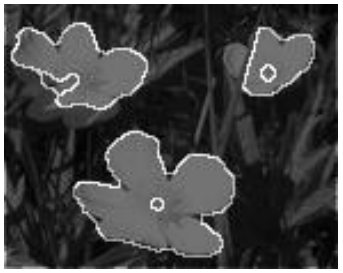
Experimentations

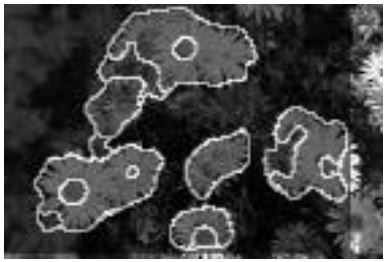
- On cars:

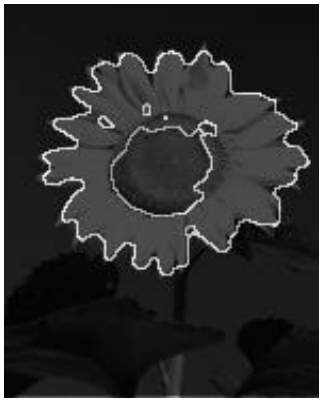




- On flowers:







Expérimentations: étiquetage d ’une image cérébrale

- Le graphe conceptuel contient 15 nœuds et 101 arcs. Dans le résultat chaque couleur correspond à une entité cérébrale spécifique (cortex, substance blanche, les différents noyaux gris internes)

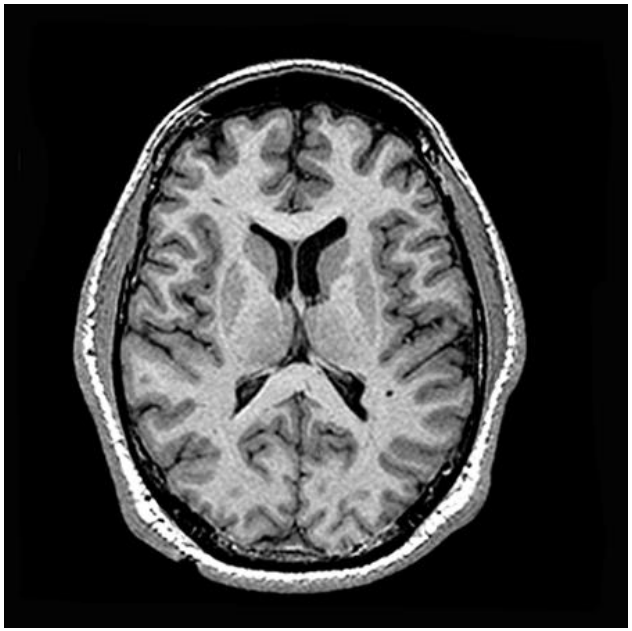


Image d’origine

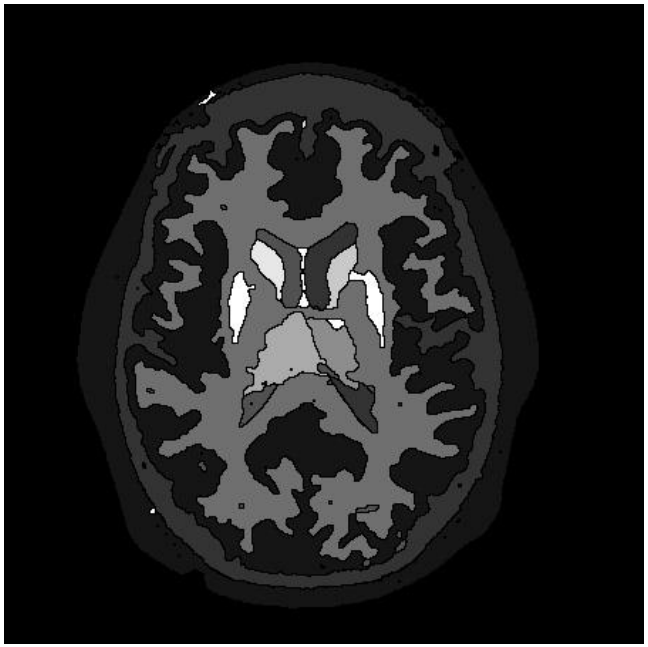


Image étiquetée

Expérimentations: étiquetage d ’une image cérébrale

- Le graphe conceptuel contient 15 nœuds et 101 arcs. Dans le résultat chaque couleur correspond à une entité cérébrale spécifique (cortex, substance blanche, les différents noyaux gris internes)

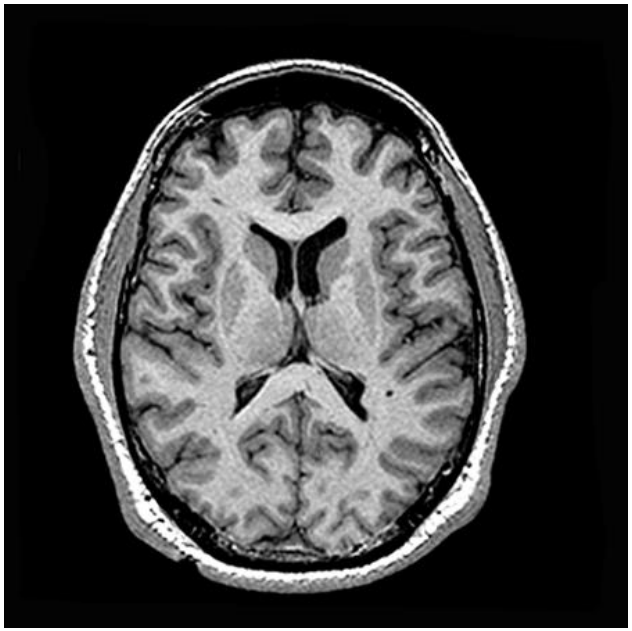


Image d’origine

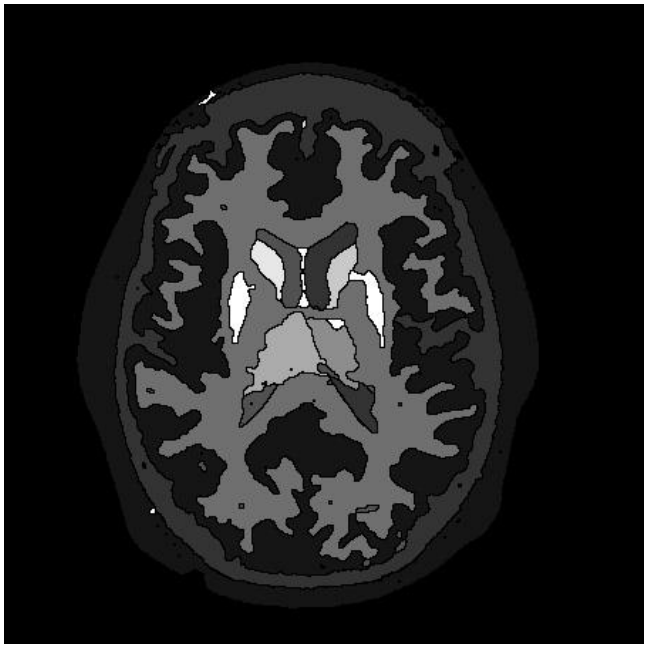


Image étiquetée

Hypergraphes

La vérification de la consistance d'arc interdit la possibilité de retrouver des organisations spatiales faisant intervenir plus de 2 entités. Il n'est pas possible par exemple, de retrouver des alignements d'objets, de vérifier si on passe par un nœud donné pour aller d'un nœud à un autre.

- Solution: exprimer ces relations avec des hyper-arcs.
- Problème: complexité plus grande

Hypergraphes

- Général : une hyper-arête = un ensemble de nœuds. Forcément non-orientée.

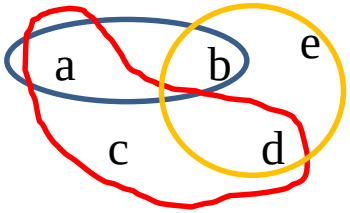
Définition: Un hyper-graphe H = (V , E).

V =ensemble des sommets de H

E est une partie de P(V).

Un élément de E est appelé hyper-arête.

Un hyper-graphe H = (V , E) est fini si V est fini.



Exemple d'hyper-graphe non orienté

Soient V = {a, b, c, d , e} et

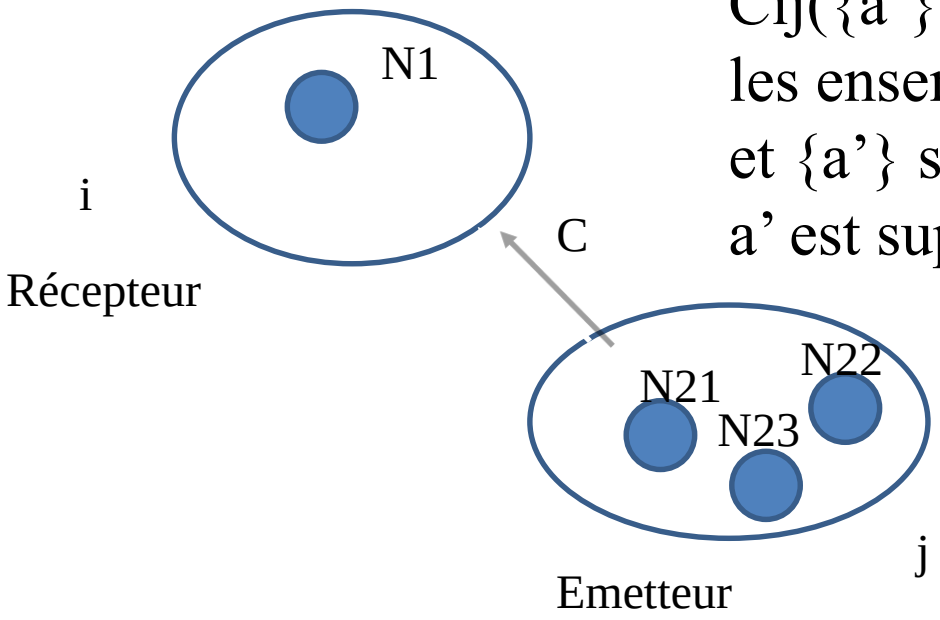
E = {{a, b}, {a, c, d }, {b, d , e}}.

- Orienté : un hyper-arc = une paire ordonnée de deux ensembles de nœuds (un ensemble émetteur et un ensemble récepteur).

Ici on utilisera la notion d'hypergraphe orienté

Hypergraphes

Introduction dans le cadre de la vérification de la consistance d'arc. Possibilité d'avoir des relations n-aires



Cij({a'} ,{a, b, c}) est vraie si
les ensembles d'étiquettes {a, b, c}
et {a'} satisfont la contrainte C.
a' est supporté par l'ensemble {a,b,c}.

Consistance d'hyper-arc

Le principe est le même que pour les contraintes binaires : une contrainte est Hyper Arc Consistante si et seulement si chaque valeur de chaque variable appartient à une solution de la contrainte. On établit la consistance d'hyper-arc en supprimant toutes les valeurs qui ne satisfont pas cette propriété.

Définitions:

Soit un CSP (X,D,C), et une contrainte $c \in C$ portant sur les variables x1, x2,...,xn avec leur domaine respectif $D_{x1}, D_{x2}, ..., D_{xn}$ de sorte que $c \in D_{x1} \times D_{x2} \times ... \times D_{xn}$. On dit alors que c est hyper-arc consistante si pour chaque $i \in [1..n]$ et $a \in Dxi$, il existe un n-uplet d dans c avec $a \in d$ et d vérifie la contrainte c.

Un CSP est hyper-arc consistant si toutes ses contraintes sont hyper-arc consistantes.

Consistance d'hyper-arc à deux niveaux de contraintes

- Soit un CSP (X,D,C), et une contrainte c ∈ C portant sur les noeuds x1, x2,...,xn avec leur domaine respectif D_{x1},D_{x2},...,D_{xn} de sorte que c ⊆ D_{x1} ×D_{x2} ×... ×D_{xn}.

- Soit Cmp_{xi} une relation de compatibilité (contrainte intra-nœud) associée au nœud xi, i ∈ [1 ..n] telle que (a,b) ∈ Cmp_{xi} ssi a et b sont compatibles.

On dit alors que c est hyper-arc consistante à deux niveaux de contrainte si pour chaque i ∈ [1..n] et a ∈ Dxi, ∃ a' ∈ Dxi et ∃ un n-uplet d dans c avec a ∈ d et (a,a') ∈ Cmp_{xi} et d vérifie la contrainte c.

Complexité et gestion du temps de calcul

Complexité : $O(d^k)$, $k<=n$ en fonction de la taille de l'hyper-arc.

Réduction de la taille du domaine de recherche par :

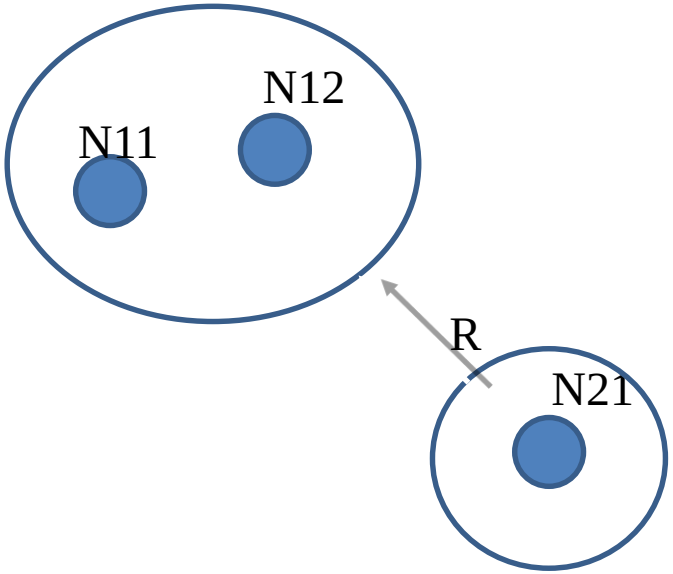
partitionnement des données.

Ajout itératif d'arcs dans la vérification de la consistance d'arc.

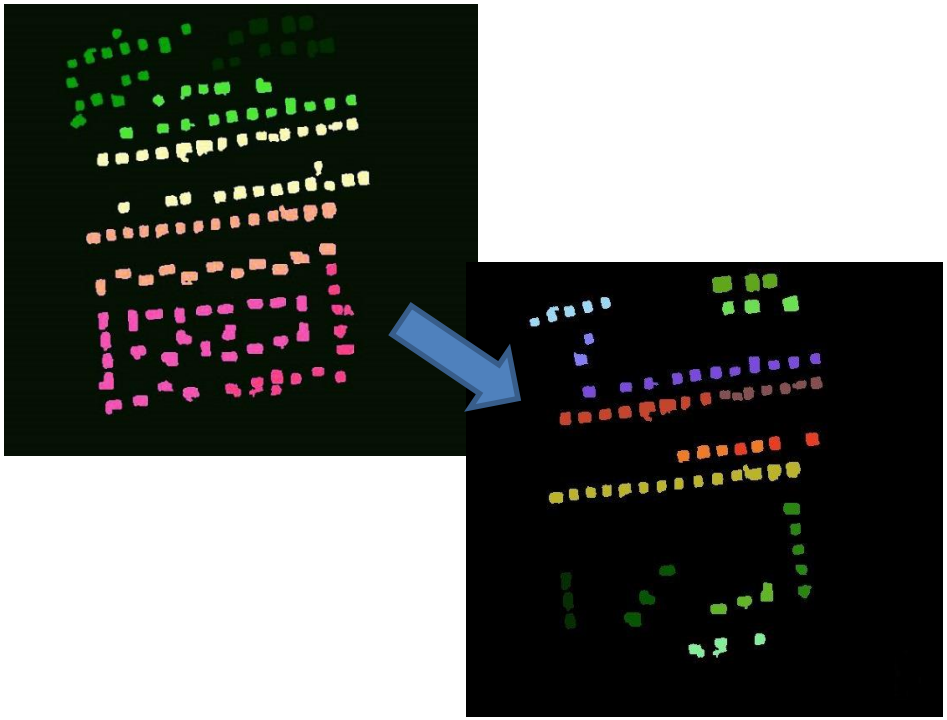
Hypergraphes

Détecter des alignements

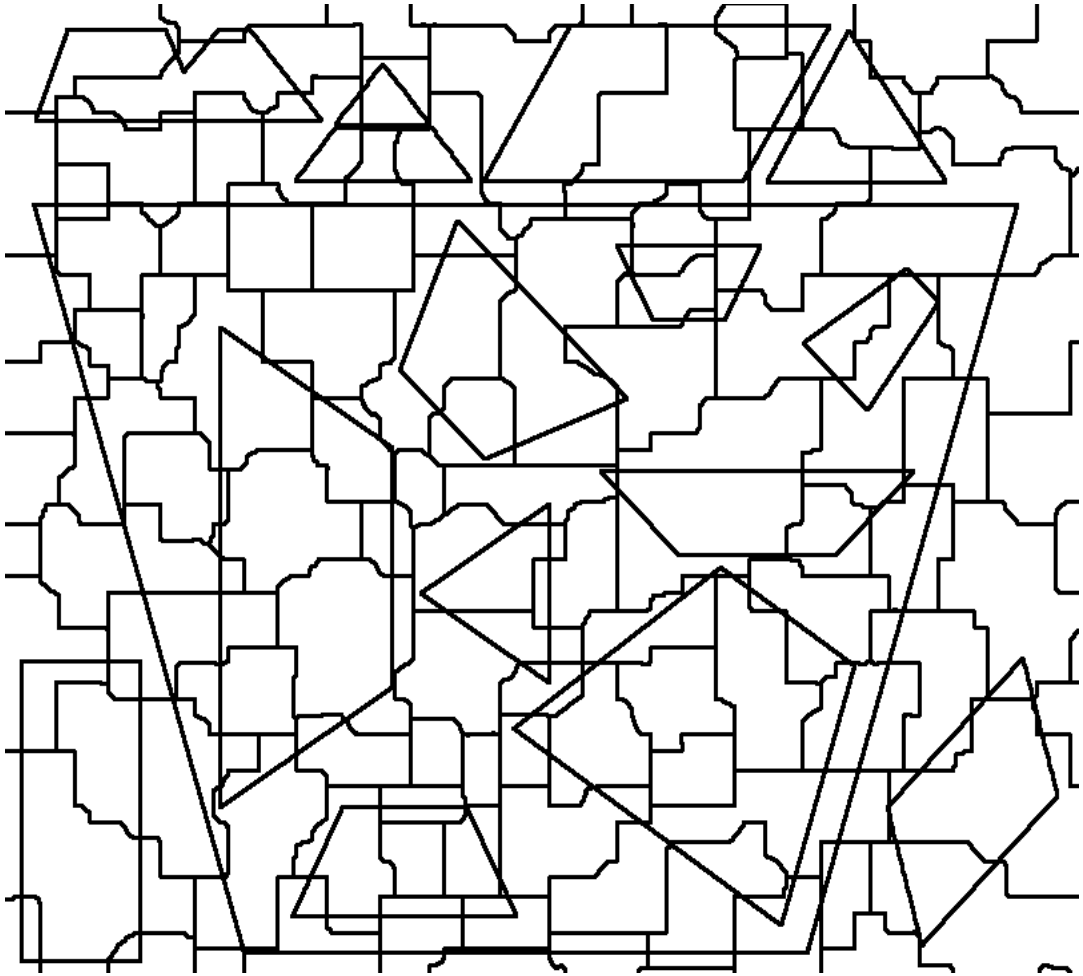
R= N11 et N12 sont alignés avec N21



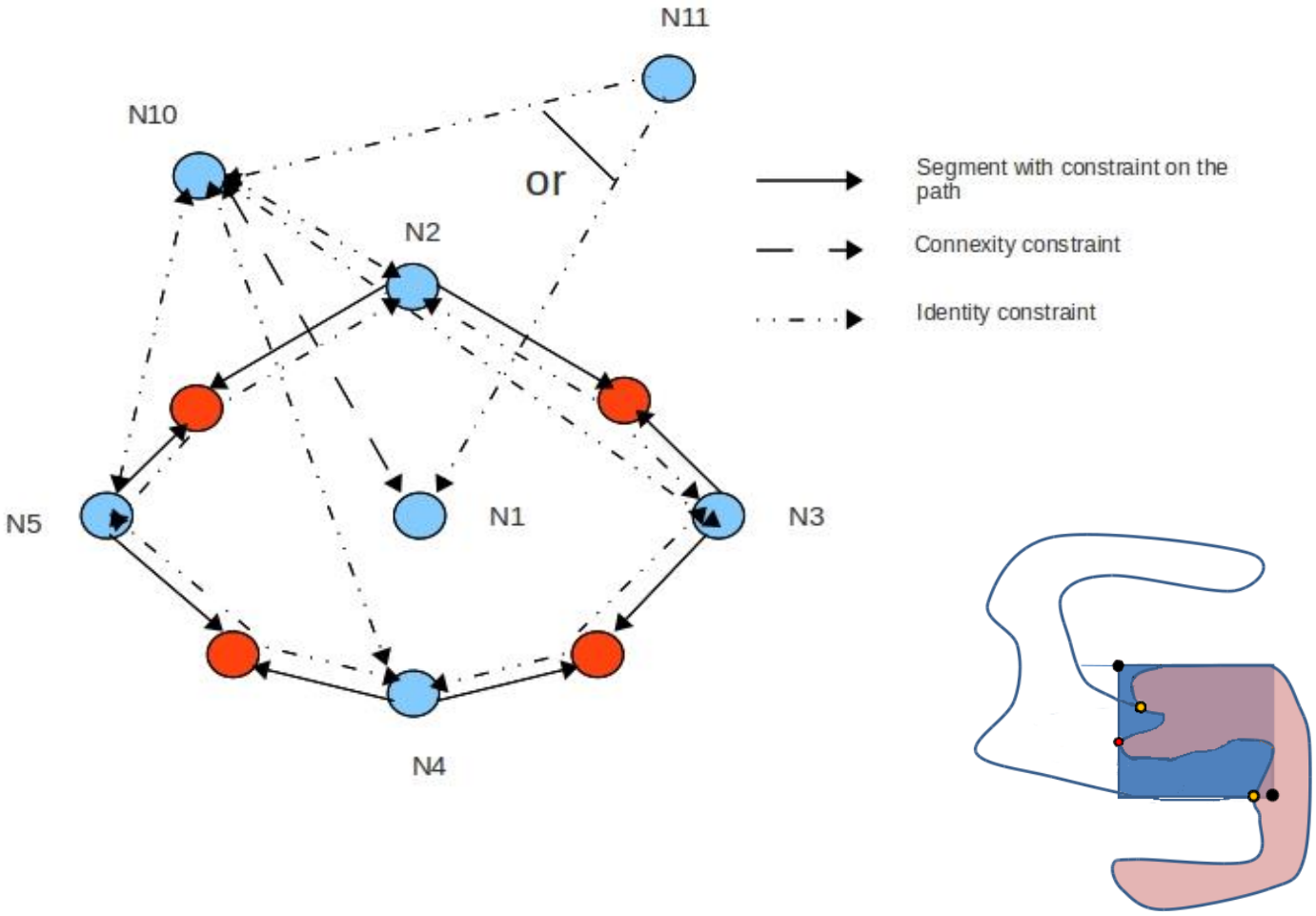
Application à la détection d'alignements dans les lotissements.



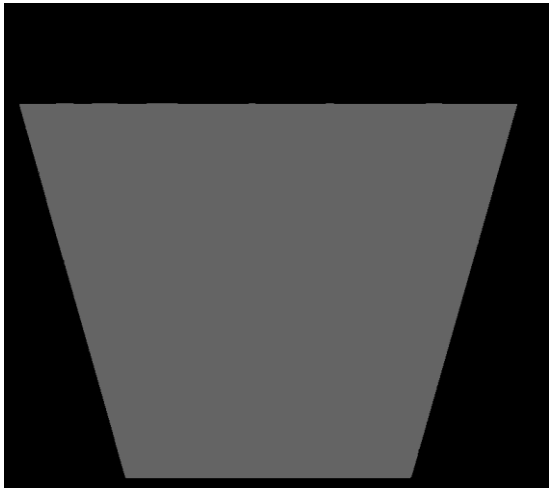
Application à des formes géométriques

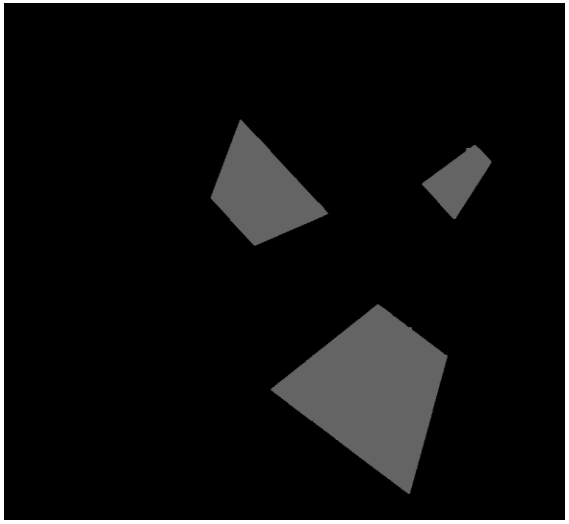


Exemple de graphe décrivant un trapèze

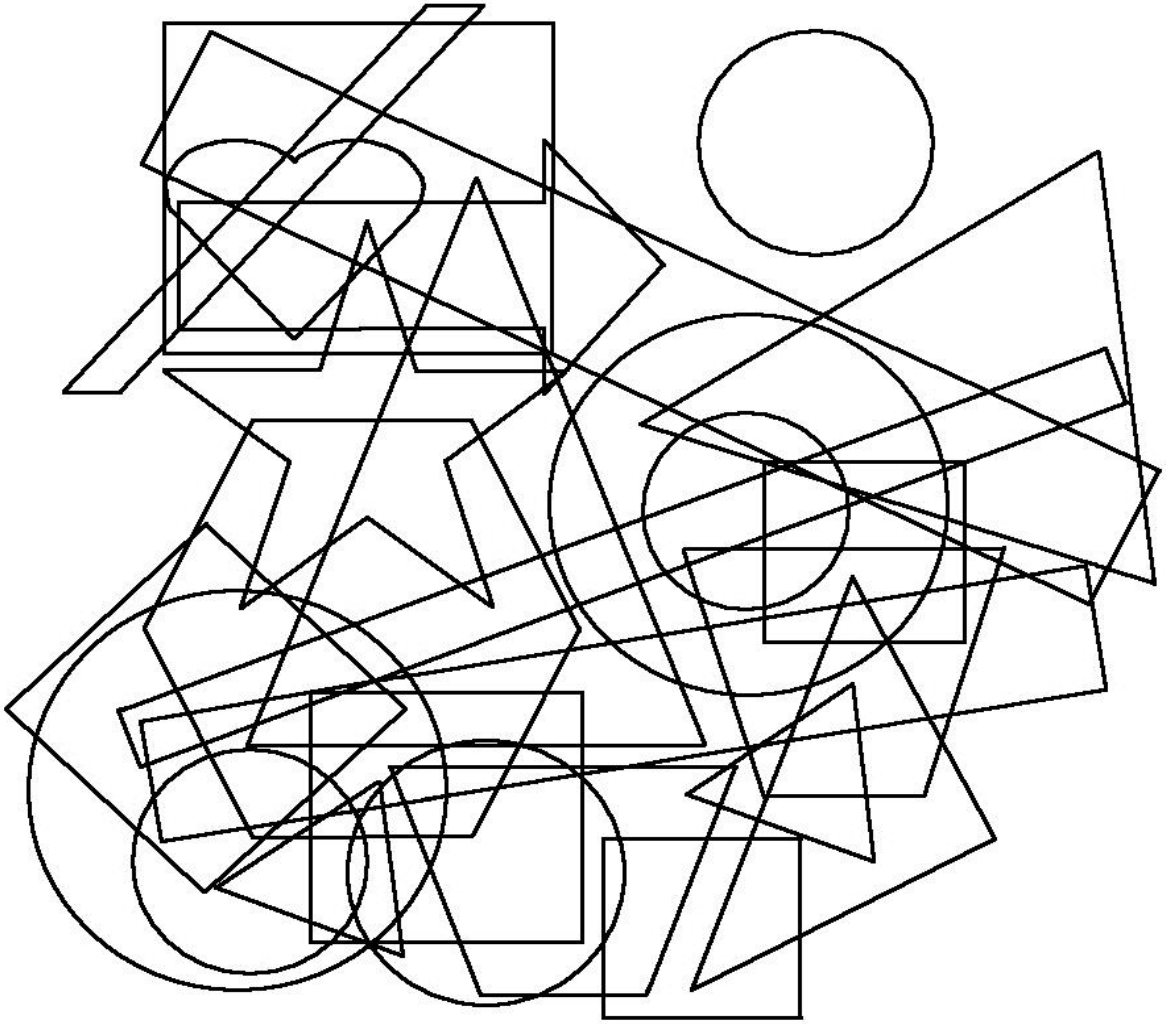


Application sur des formes géométriques

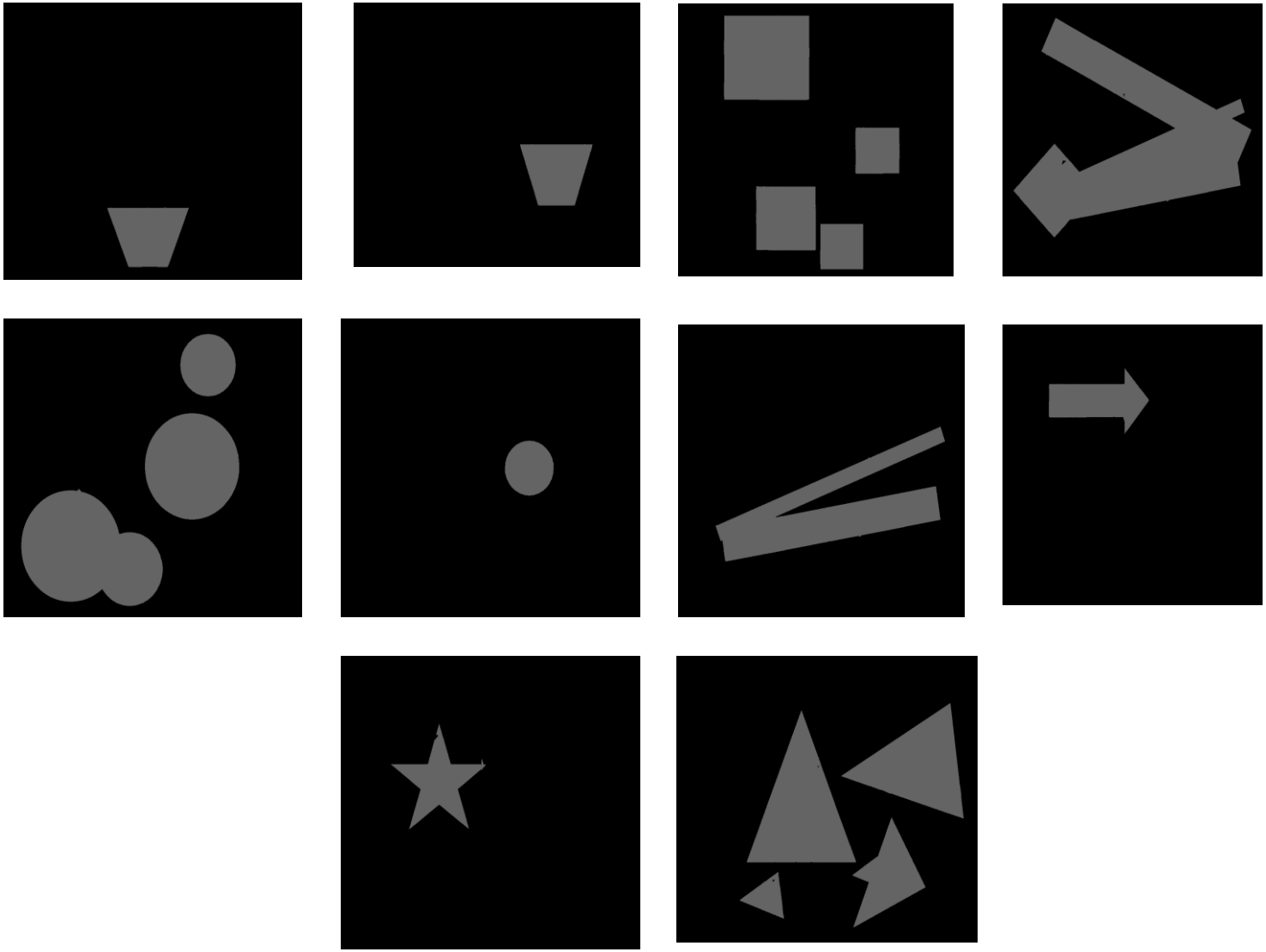




Autre exemple



Autre exemple



Extraction du corps calleux

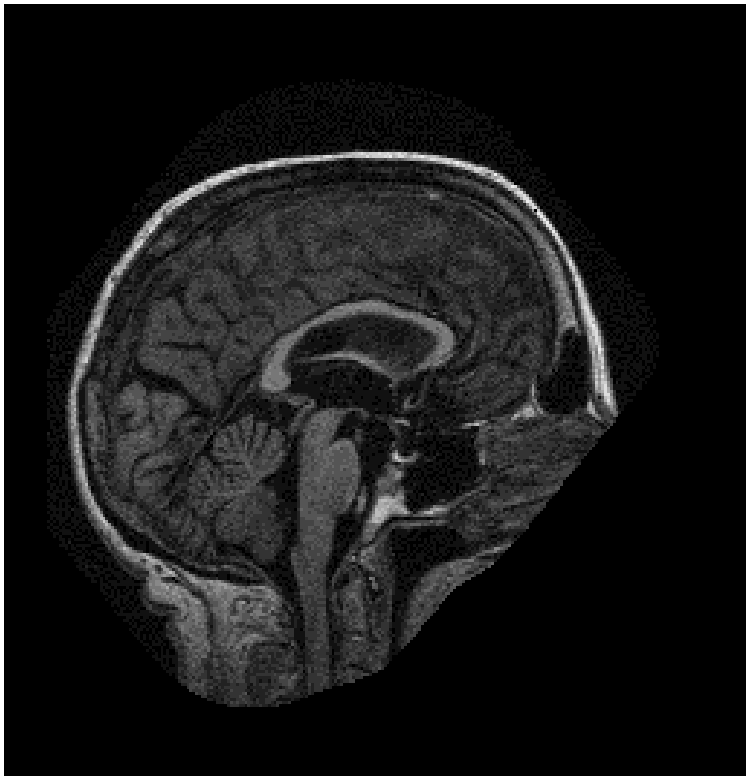


Image d'origine

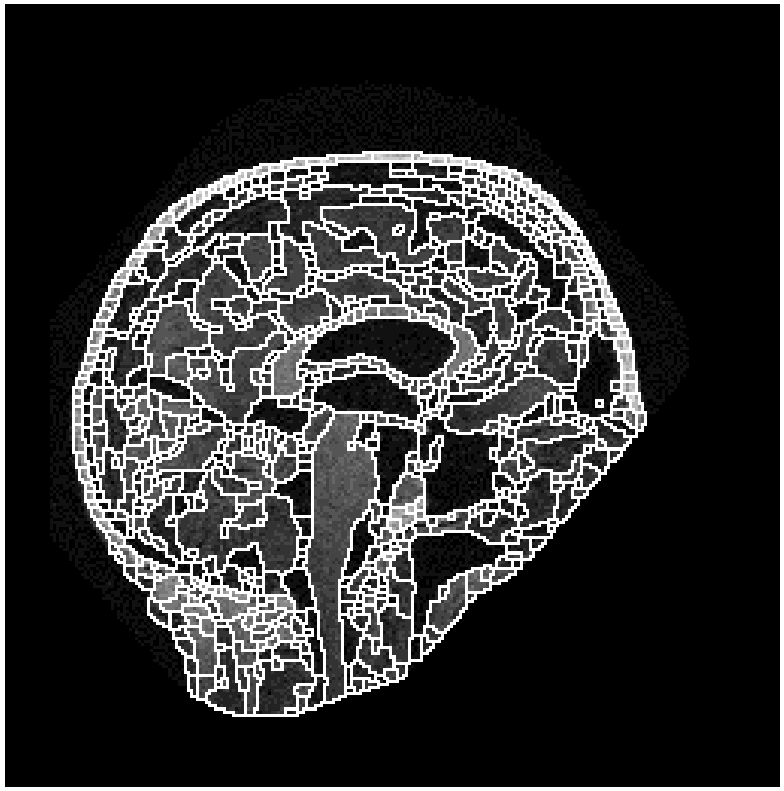
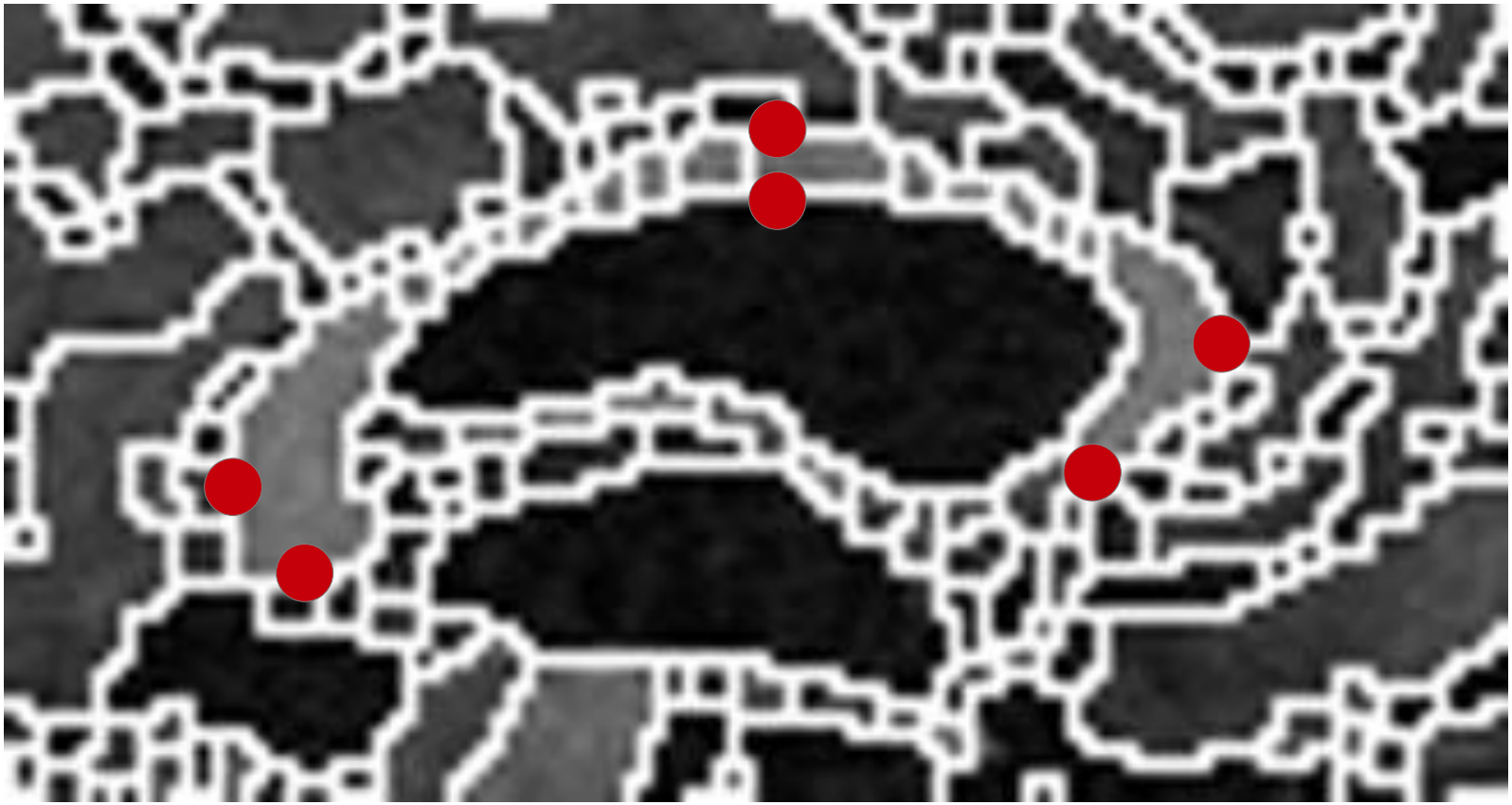


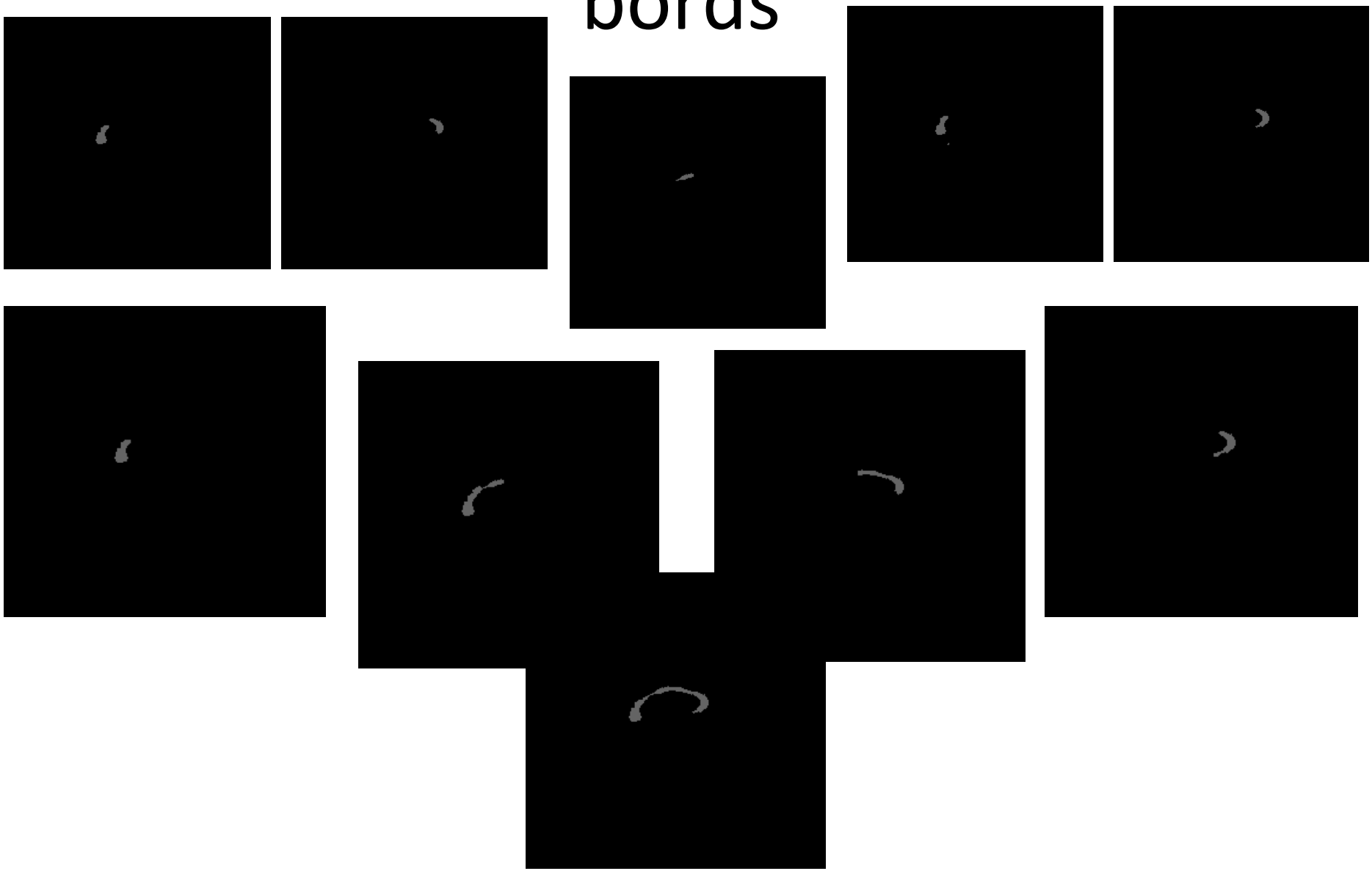
image segmentée par un watershed

Extraction du corps calleux

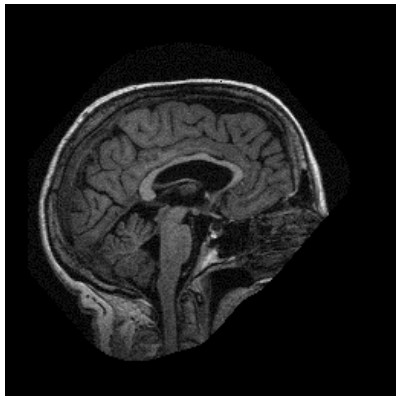
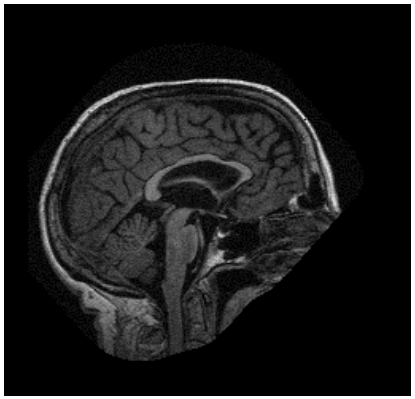
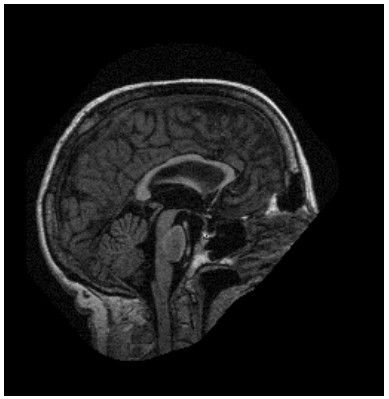
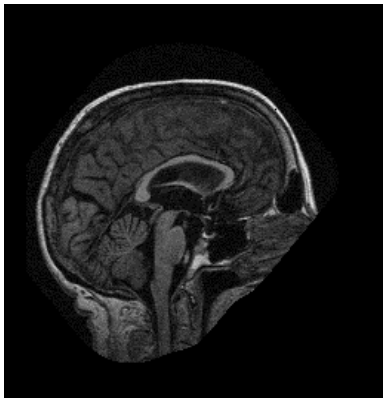
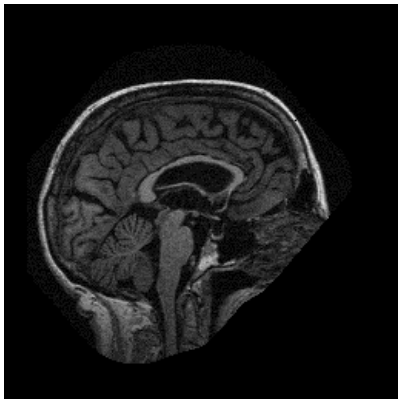
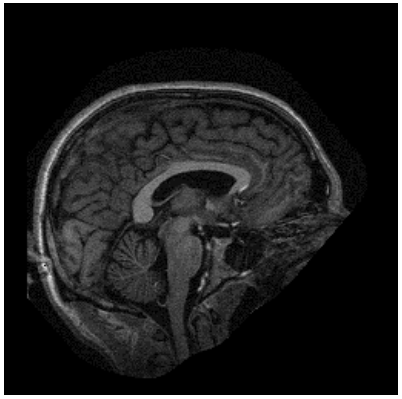
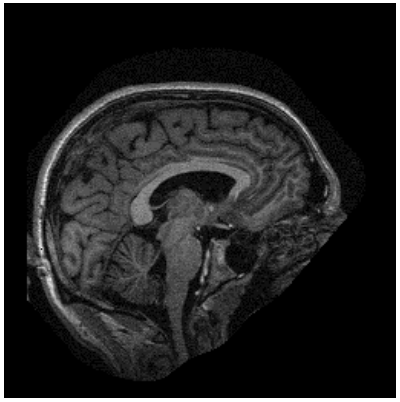


che des extremums en fonction de contraintes de distance et de chemin entre chacun d'ent

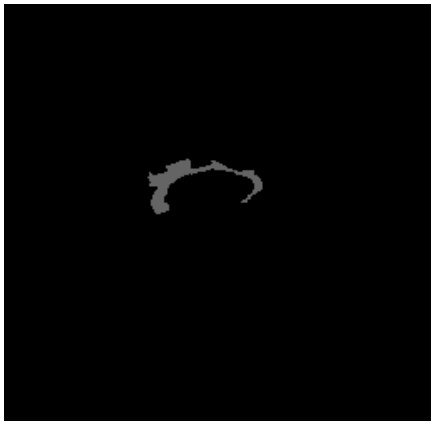
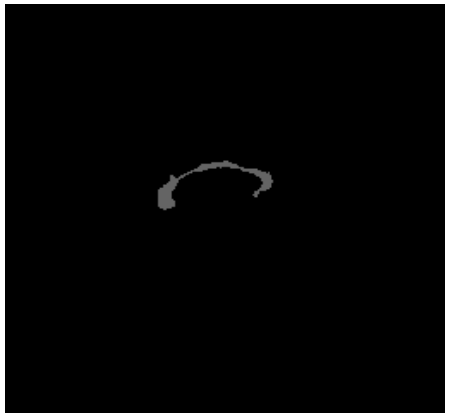
Extraction des extremums et des bords

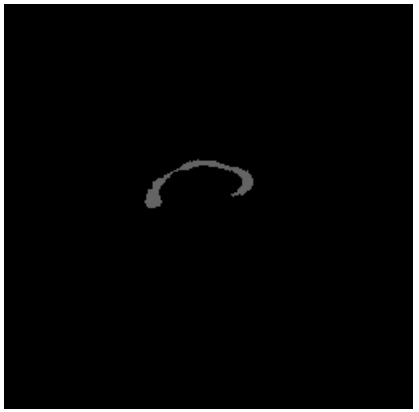
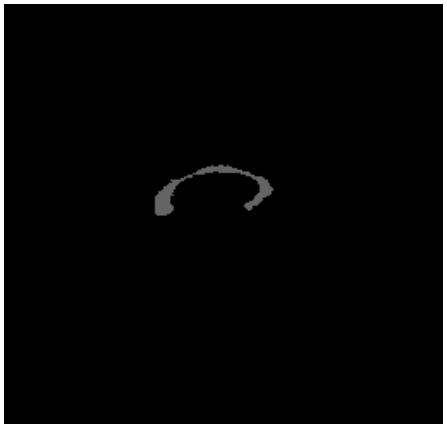
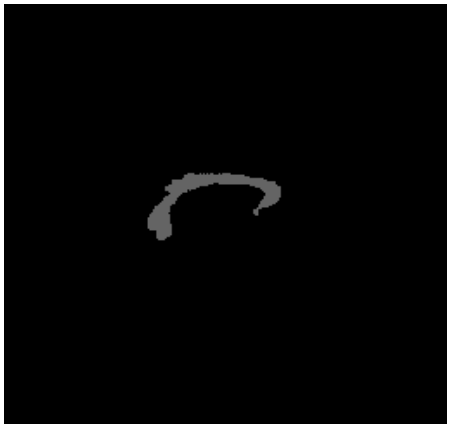
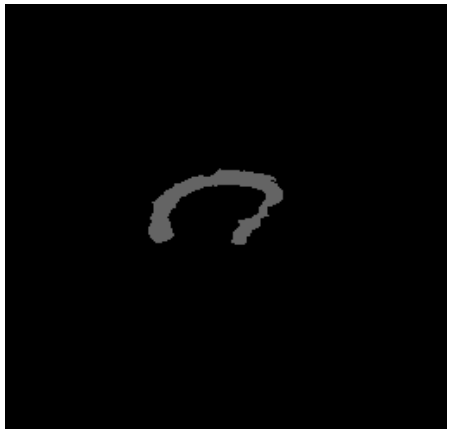


Extraction du corps calleux

Extraction du corps calleux

Plan

- Pourquoi les graphes ?
- Rappels et définitions
- De l'image à l 'objet graphe: les graphes d'adjacence de régions
- Simplification de graphes
  - division et fusion de régions
  - fusion pyramidale
- Interprétation d 'images: Mise en correspondance de graphes

  Isomorphisme de graphes et sous graphes

  Isomorphisme de sous-graphe avec tolérance d 'erreur

  Mise en correspondance inexacte: un problème de satisfaction de contraintes
- Classification de graphes : Noyaux sur Graphes

Noyaux sur Graphes : motivation

Reconnaissance des formes structurelle

- Riche description des objects
- Propriétés pauvres des espaces de graphes qui ne permettent pas de généraliser/combiner les ensembles de graphes.

Reconnaissances des formes statistiques

- Description globales des objects
- Espaces numériques avec de nombreuses propriétés mathématiques (metrique, espaces vectoriels, . . .).

Motivation

Analyser de grandes familles d'objets structurels et numériques en utilisant

un cadre unifié basée sur la similarité par paires

Plan

1- Bases sur les noyaux

2- Noyaux sur graphes

Noyaux basés sur des sacs infinis

Noyaux basés sur des sacs finis

- Sacs de chemins
- Noyaux sur des sacs de chemins
- Noyaux sur des chemins

Definition

- A kernel k is a **symmetric** similarity measure on a set χ

$$\forall (x, y) \in \chi^2,\ k(x, y) = k(y, x)$$

- k is said to be **definite positive** (d.p.) iff k is symmetric and iff:

$$\left.\begin{array}{rcl} \forall (x_1, \ldots, x_n) & \in & \chi^n \\ \forall (c_1, \ldots, c_n) & \in & \mathbb{R}^n \end{array}\right\} \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$$

- $K = (k(x_i, x_j))_{(i,j) \in \{1,\ldots,n\}}$ is the Gramm matrix of k. k is d.p. iff:

$$\forall c \in \mathbb{R}^n - \{0\},\ c^t K c \geq 0$$

Examples

If $\chi = \mathbb{R}^n$, classical kernels include:

- Linear kernel:

$$K(x, y) = x^t y$$

- Polynomial kernel

$$K(x, y) = (x^t y)^d + c,\ c \in \mathbb{R}, d \in \mathbb{N}$$

- Cosinus kernel:

$$K(x, y) = \frac{x^t y}{\|x\|\|y\|}$$

- Rational kernel:

$$K(x, y) = 1 - \frac{\|x - y\|^2}{\|x - y\|^2 + b}, b \in \mathbb{R} - \{0\}$$

- Gaussian Kernel

$$K(x, y) = exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right), \sigma \in \mathbb{R} - \{0\}$$

Kernels and scalar products

Aronszajn 1950 :

A kernel k is definite positive on a space χ
if and only if
it exists

- one Hilbert space $\mathcal{H}$ and
- a function $\varphi : \chi \rightarrow \mathcal{H}$

such that:

$$k(x, y) =< \varphi(x), \varphi(y) >$$

The Kernel Trick

- Linear classifier become non-linear using kernels

- Problem: φ is usually unknown.
- Many methods only need scalar product between data (not explicit coordinates) $\Rightarrow$ replace scalar product by kernel.
- E.g. k-NN:

$$\begin{aligned} d_K^2(x_1, x_2) &= \|\varphi(x_1) - \varphi(x_2)\|^2 \\ &= <\varphi(x_1) - \varphi(x_2), \varphi(x_1) - \varphi(x_2)> \\ &= <\varphi(x_1), \varphi(x_1)> + <\varphi(x_2), \varphi(x_2)> -2<\varphi(x_1), \varphi(x_2)> \\ d_K(x_1, x_2) &= k(x_1, x_1) + k(x_2, x_2) - 2k(x_1, x_2) \end{aligned}$$

- Kernel trick
  - Algorithm defined in $\mathcal{H} \Rightarrow$ (linear methods,non linear separation),
  - Data stored in χ.

Interesting but so what...

Le « kernel trick »

Noyaux et données structurées :
Le « kernel trick » fournit un plongement implicite dont la métrique est définie à partir de notre critère de similarité (Le noyau).

Plan

1- Bases sur les noyaux

2- Noyaux sur graphes

Noyaux basés sur des sacs infinis

Noyaux basés sur des sacs finis

- Sacs de chemins
- Noyaux sur des sacs de chemins
- Noyaux sur des chemins

Noyaux sur graphes

Etape 1: Construction d'un graphe

Méthode 1: calcul du graphe d'ajacence à partir d'une images segmentée.

Methode 2: Recherche du squelette morphologique à partir de l'image binaire.

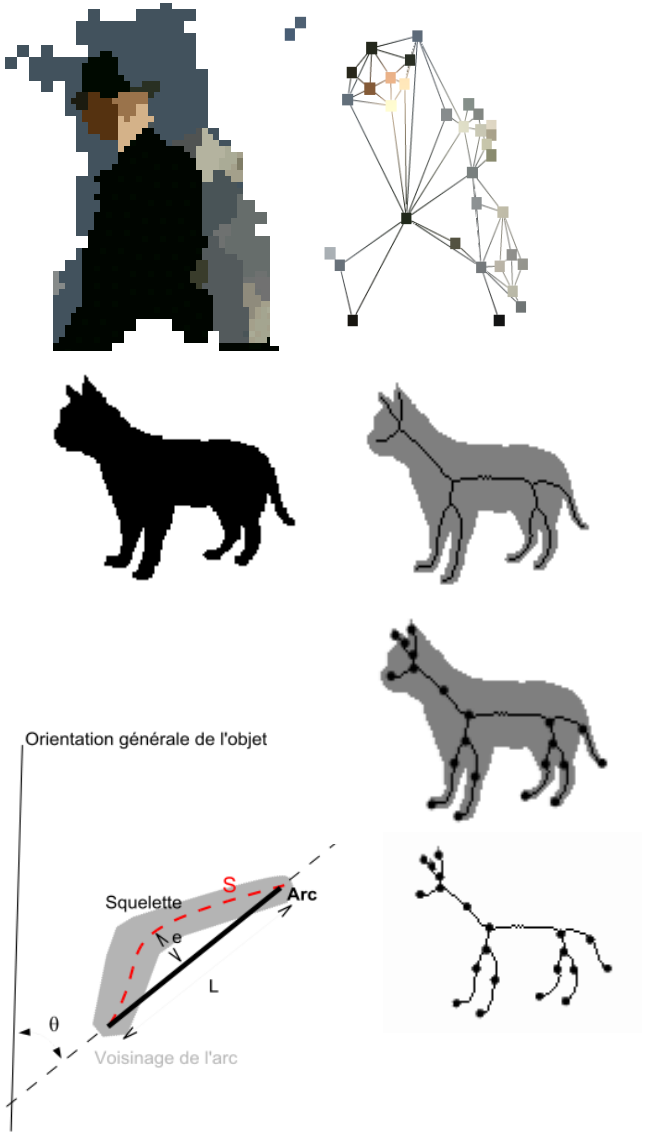
Etape 2: Étiquetage du graphe. Apport d'information concernant l'image définie par le graphe.

Pour les nœuds :

épaisseur de l'objet à l'emplacement du nœud, coordonnées du nœud, information de voisinage du nœud : moyenne des niveaux de gris, variance, texture, histogramme.

Pour les arcs :

longueur de l'arc (L), longueur de la branche du squelette (s),orientation (θ) de l'arc, aire (A) du voisinage, écart (e) maximal entre l'arc et la branche du squelette.



Noyaux sur graphes

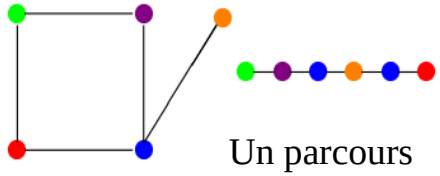
Classifieur SVM (Support Vector Machine)

Définition de la frontière du classifieur en fonction du produit scalaire des points d'apprentissage.

Pour les graphes, définition d'un noyau, i.e. un produit scalaire entre graphes :

- K(Gi , Gj) = <Gi , Gj>

Idée: comparer les parcours présents dans chaque graphe et les étiquettes présentes sur le parcours.



2 Problèmes:

- Choix des parcours à comparer

Chemins aléatoires : prise en compte de tous les parcours possibles.

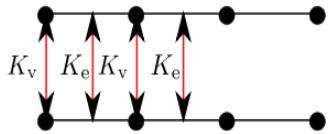
Alternatives: garder un petit pourcentage des parcours les plus pertinents

- Parcours les plus courts d'un nœud à l'autre (méthode des k-parcours).
- Élimination itérative des parcours moins pertinents tout en gardant la couverture du graphe

- Choix du noyaux K

Exemples noyaux :

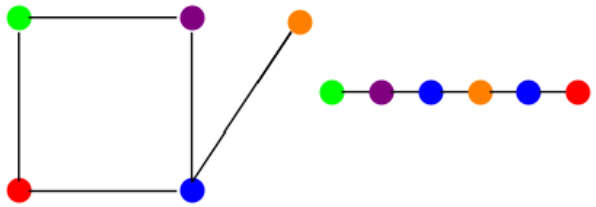
- Alignement de parcours [Kashima et al, 2002]



- Noyaux d'édition sur chemins [Luc Brun, et al, 2011]

Walk kernels

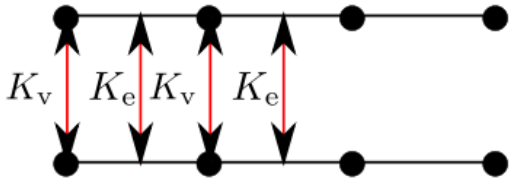
- Walks: Let $G = (V, E)$. $W = (v_1, \ldots, v_n)$ is a walk iff $(v_i, v_{i+1}) \in E, \forall i \in \{1, \ldots, n-1\}$.



is a walk

- Kernel between walks

$$K(h, h') = \begin{cases} 0 & \text{if } |h| \neq [h'| \text{ and} \\ K_v(v_1, v'_1).\Pi_{i=1}^{|h|} K_e(e_i, e'_i) K_v(v_{i+1}, v'_{i+1}) & \text{otherwize} \end{cases}$$



Noyaux sur Chemins

- Walk kernels :

$$K(G_1, G_2) = \sum_{h \in \mathcal{W}(G_1)} \sum_{h' \in \mathcal{W}(G_2)} K(h, h') \lambda_{G_1}(h) \lambda_{G_2}(h')$$

- Covers different Graph kernels[Vert 2007,Vishwanathan et al. 2010]:

$$If\ \lambda_G(h) = \begin{cases} 1\ iff\ |h| = n & K \text{is a nth order walk kernel} \\ P_G(h)(Markov\ RW) & K \text{is a random walk/marginalized ker} \\ \beta^{|h|} & K \text{is a geometric kernel} \end{cases}$$

$$P_G(h) = p_s(h_1) \Pi_{i=1}^{n} p_t(h_i | h_{i-1}) p_q(h_n) \text{ with } |h| = n$$

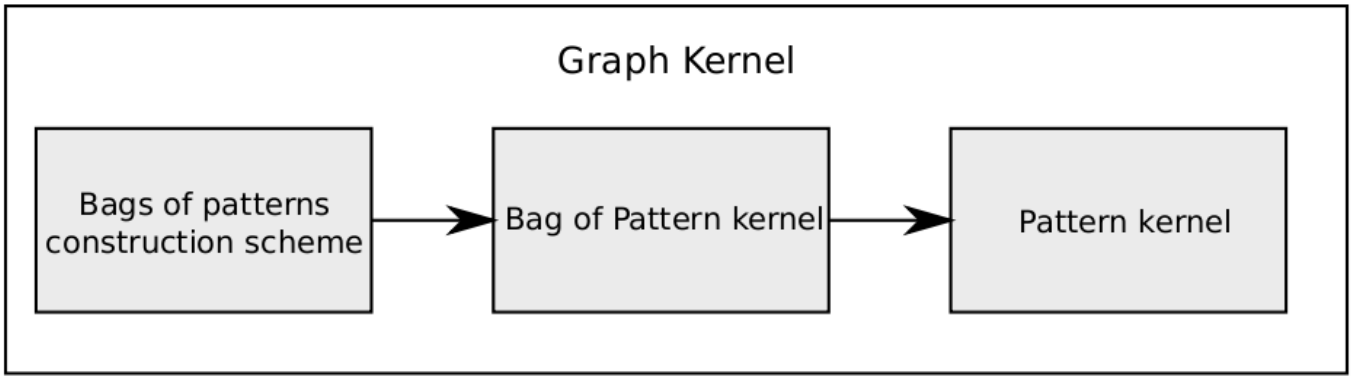
Les chemins peuvent induire des problèmes d'instabilité: Des chemins avec des longueurs arbiraires sur un même ensemble d'arcs et de noeuds.

Cadre étendu à des arborescences (tree-pattern) [Vert 2006, Bach 2007]

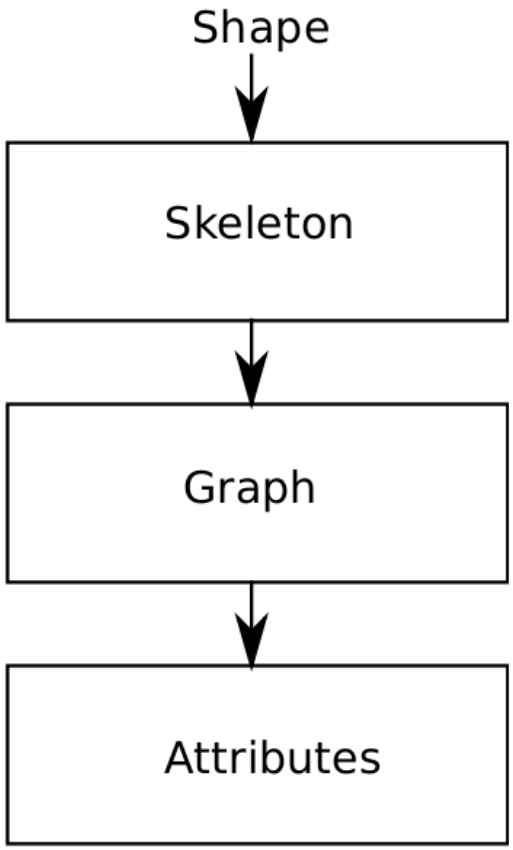
Finite Bag kernels

$$\left.\begin{array}{rcl} G & \rightarrow & B(G) \\ G' & \rightarrow & B(G') \end{array}\right\} K(G,G') = K(B(G),B(G'))$$

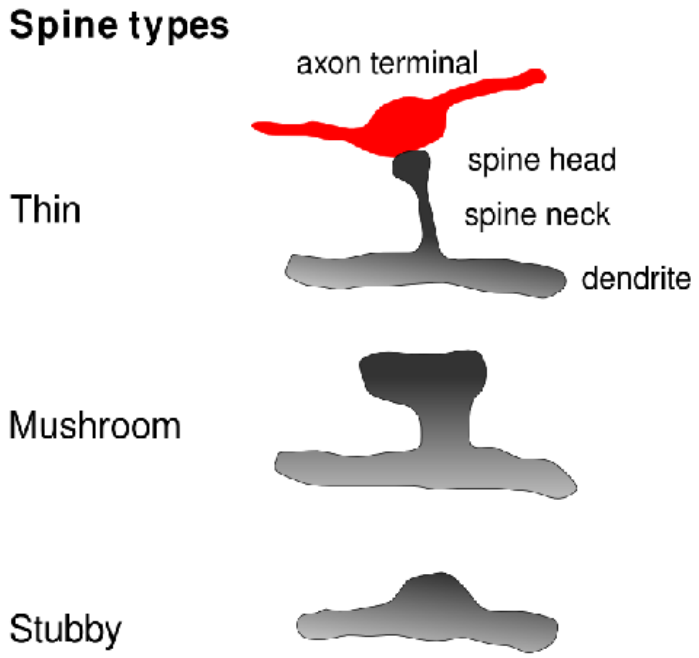
Three independent step to design a graph kernel.



From Shapes to Graphs

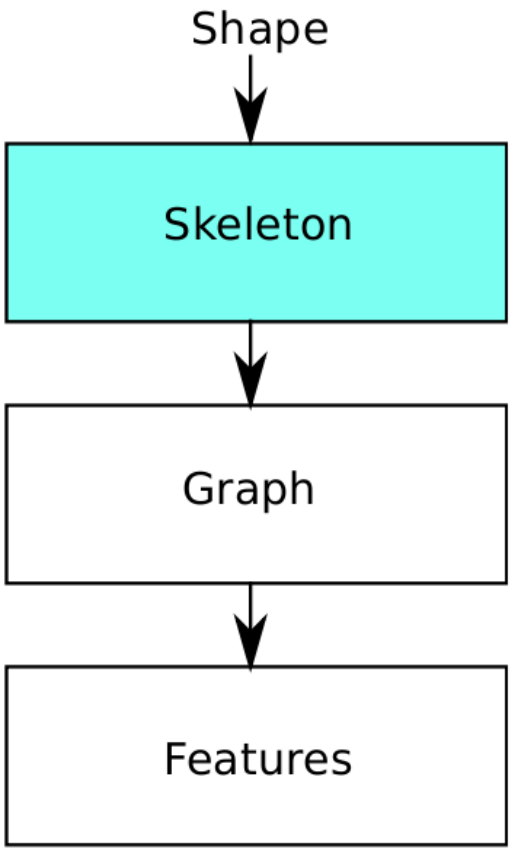


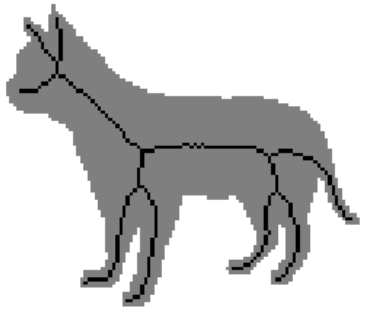
Motivation : Analyse the shape of Dentritic's spines



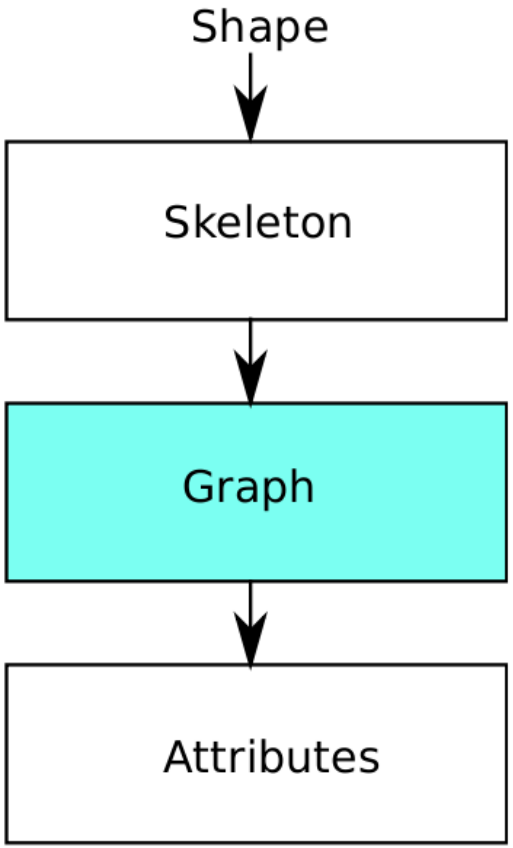
François Xavier Dupé's PhD

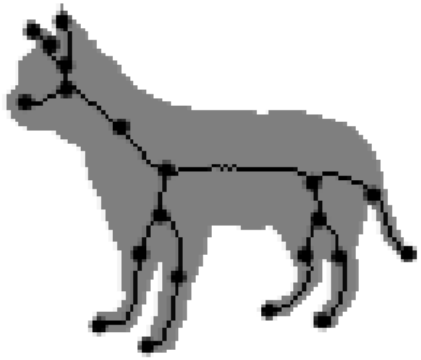
From Shapes to Graphs





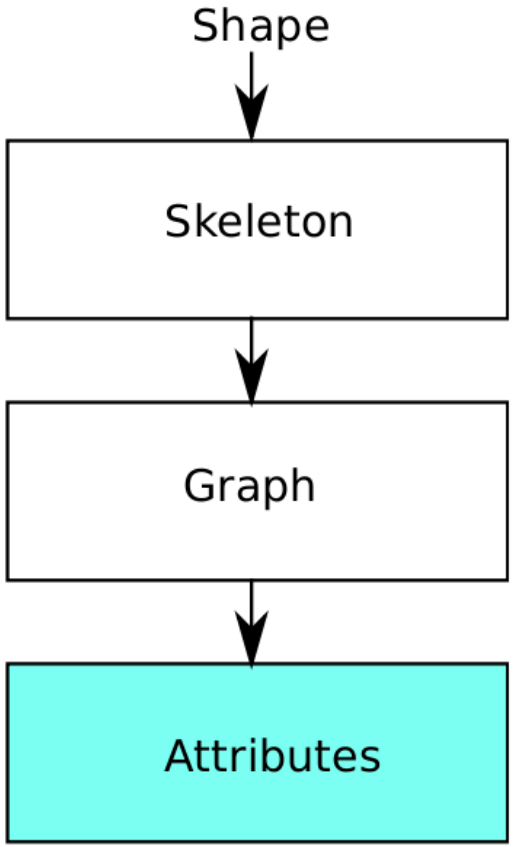
From Shapes to Graphs





- V:
  - extremities/intersection of branches or
  - abrupt changes of the radius along a branch.
- E:
  - branches of the skeleton.

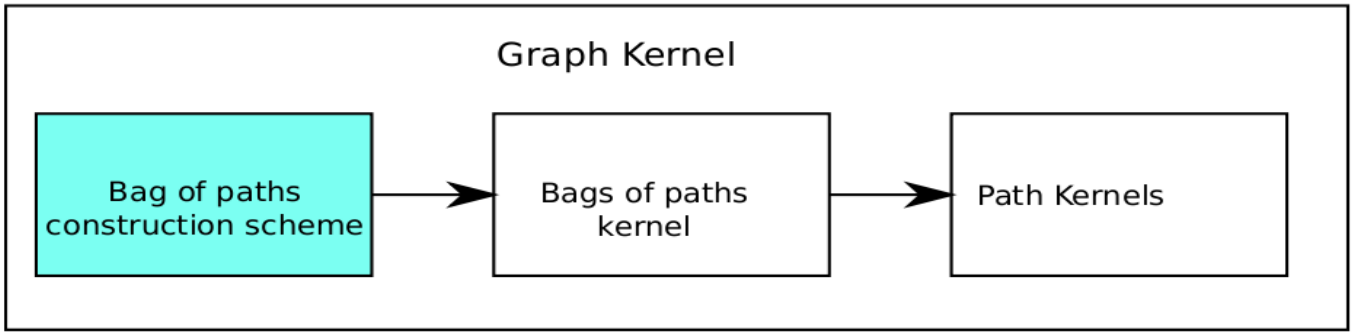
From Shapes to Graphs

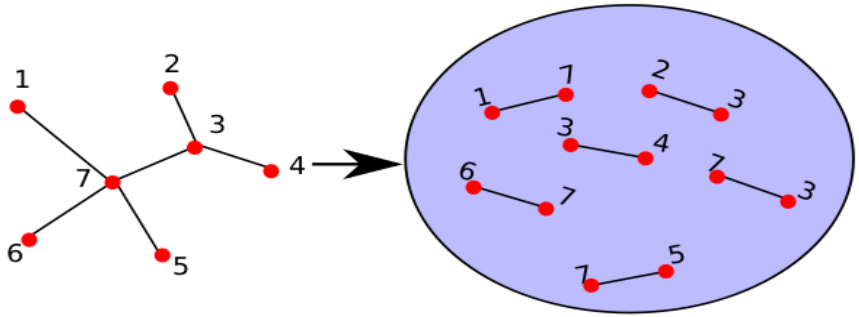


- V:
  - Distance to the barycenter of the shape
- E:
  - Interpolation of the radius along the branch,
  - Each edge is weighted by a function ω encoding the length of the boundary which contributed to the branch [Torsello04].

Encodes edge's relevance [Torsello et Hancock, 2004].

Construction Framework





Comparer les sacs de chemins : estimer la similarité entre les graphes

Sacs de chemins : Méthodes existantes

- tous les chemins [Kashima et al, 2002 ; Borgwardt et al, 2005] ;
- l'ensemble des chemins élémentaires formés par le chemin le plus court entre chaque paire de sommets [Borgwardt et al, 2005] ;
- tous les chemins élémentaires jusqu'à une certaine longueur [Suard et al, 2006].

Méthodes sans a priori sur les objets représentés par les graphes.

Sacs de chemins : Sélection des chemins

Première méthode : garder un faible pourcentage des chemins les plus pertinents.

- Facile à implémenter 🙂 ;
- Rapide 🙂.

- Des parties de la forme peuvent être perdues 😕 ;
- Redondance de l'information 😕.

Sacs de chemins : Couverture

Deuxième méthode : couvrir le graphe avec le moins de chemins possible.

Réécriture en problème d'optimisation

$$S \in \arg\min_{l \subseteq L} -\omega(l) + \lambda \text{Card}\ (l), \quad t.q.\ l \text{ couvre } G$$

$\omega(l)$: la somme des poids des chemins dans l.
Card (l) : le nombre de chemins présent dans l.
λ : paramètre d'équilibre.

- Faible redondance ;
- Pas de perte de description ;
- Problème NP-complet .

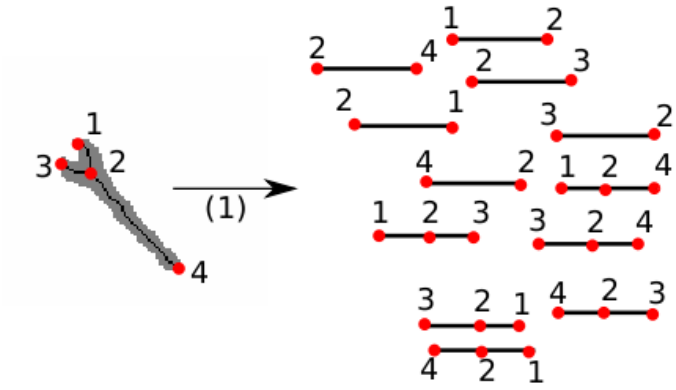
Sacs de chemins: Solution

Principe : simplifier le problème jusqu'à ce qu'il devienne résolvable

1- Supprimer itérativement les chemins les moins significatifs tout en gardant la couverture du graphe (algorithme gourmand).

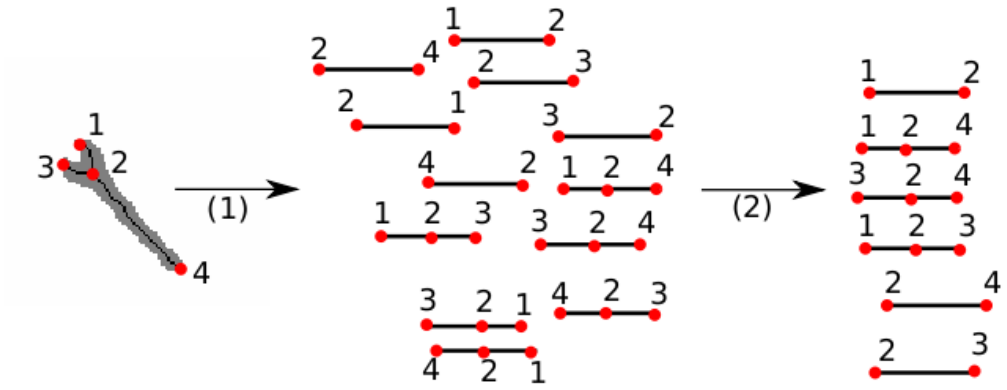
2- Utiliser le résultat de 1 comme entrée de l'algorithme de couverture de [Guo et al. 2006] (programmation dynamique).

Sacs de chemins : Exemple



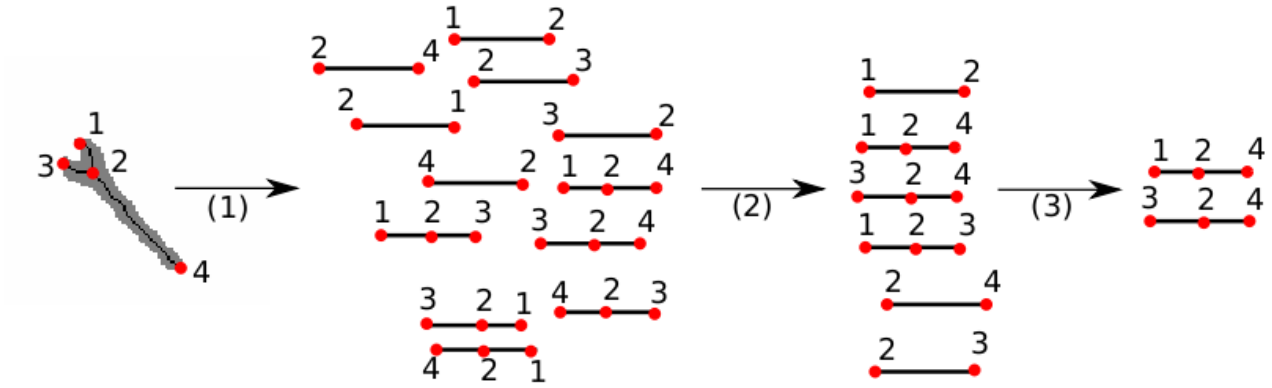
1. prendre tous les chemins jusqu'à une certaine longueur,

Sacs de chemins : Exemple



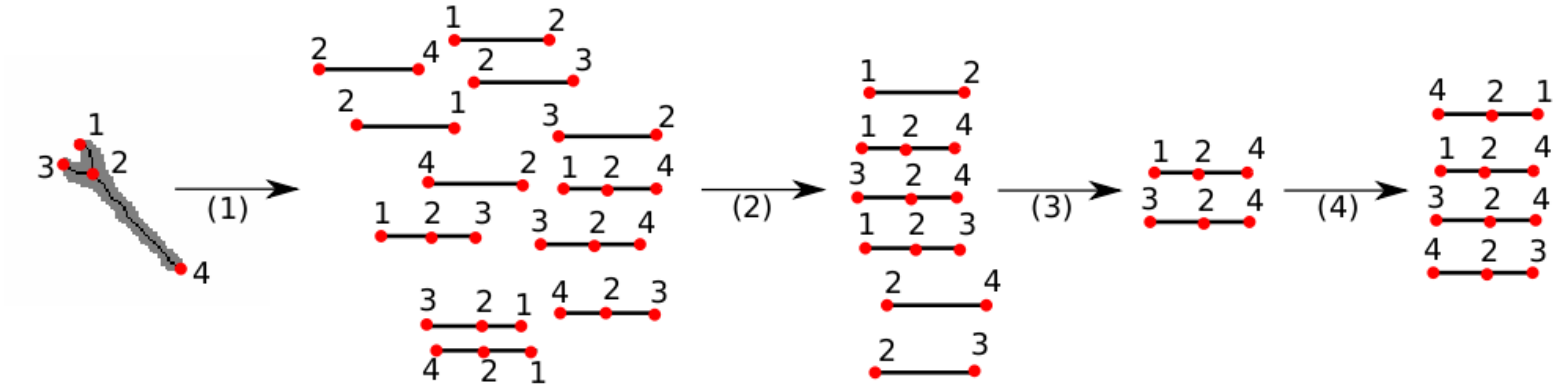
1. prendre tous les chemins jusqu'à une certaine longueur,
2. réduire la redondance,

Sacs de chemins : Exemple



1. prendre tous les chemins jusqu'à une certaine longueur,
2. réduire la redondance,
3. couverture de cardinalité minimale et de poids maximal,

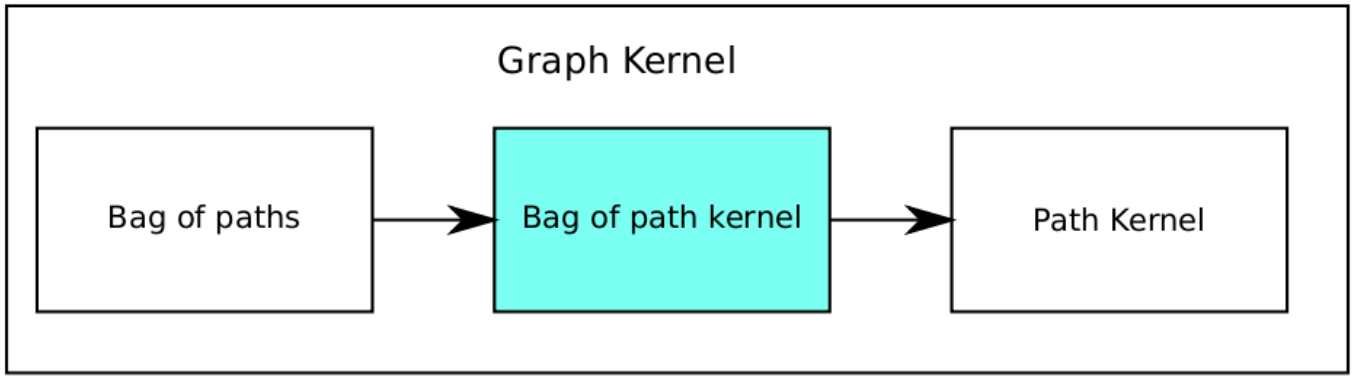
Sacs de chemins : Exemple



1. prendre tous les chemins jusqu'à une certaine longueur,
2. réduire la redondance,
3. couverture de cardinalité minimale et de poids maximal,
4. ajouter le symétrique de chaque chemin.

$\Rightarrow$ Une description complète de la forme avec seulement 4 chemins.

Kernel on Bags of Paths



Haussler99

$$K_{\text{mean}}(T_1, T_2) = \frac{1}{|T_1|}\frac{1}{|T_2|}\sum_{t\in T_1}\sum_{t\in T_2} K_{\text{path}}(t, t'),$$

More complex kernels

$$K'(x, y) = \frac{K(x,y)}{\sqrt{K(x,x)K(y,y)}}$$
$$K'(x, x) = \|\varphi'(x)\|^2 = 1$$

Normalized kernel

Weighted mean kernel

$$\begin{aligned} K_{\text{weighted}}(T_1, T_2) &= \frac{1}{|T_1|}\frac{1}{|T_2|}\sum_{t\in T_1}\sum_{t'\in T_2}\lambda_{T_1}(t), \lambda_{T_2}(t')K_{\text{path}}(t, t'), \\ \lambda_{T_i}(t) &= <\varphi(t), \mu_{T_i}>^d \end{aligned}$$

Noyaux sur sacs de chemins : Noyau moyenne pondérée

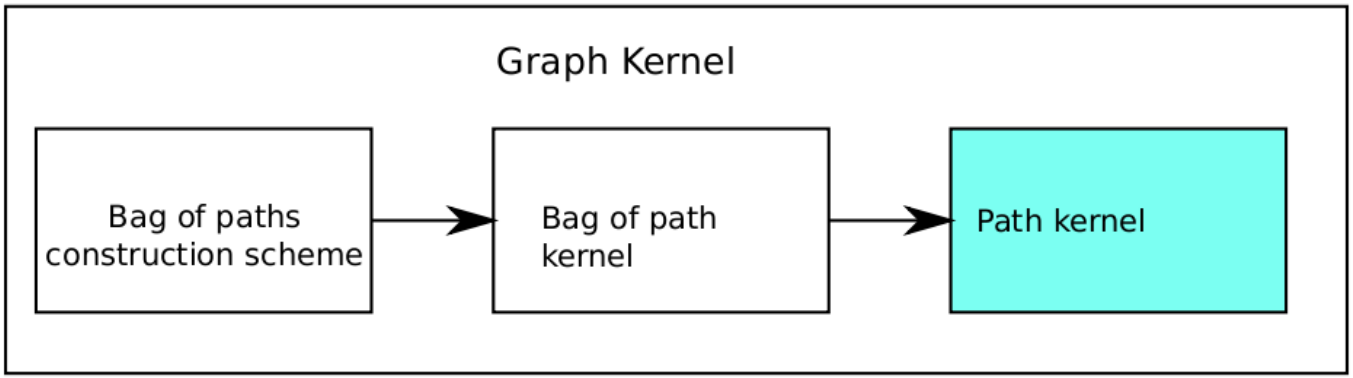
Comparaison d'ensembles $\Rightarrow$ Noyaux entre ensembles [Haussler, 1999].

Problème : la moyenne étale les résultats.
Solution : pénaliser les chemins *aberrants*.

$$K_{\text{pondere}}(T_1, T_2) = \sum_{t \in T_1} \sum_{t' \in T_2} p(t|T_1) p(t'|T_2) \langle f_{T_1}(t), f_{T_2}(t') \rangle^d \; K_{\text{chemin}}(t, t'),$$

- f : fonction de pénalisation
  (e.g. $f_T(t) = \frac{1}{|T|} \sum_{t' \in T} K_{\text{chemin}}(t, t')$) ;
- $d \in \mathbb{N}$: poids de la pénalisation ;
- $p(t|T)$: probabilité d'avoir t sachant T.

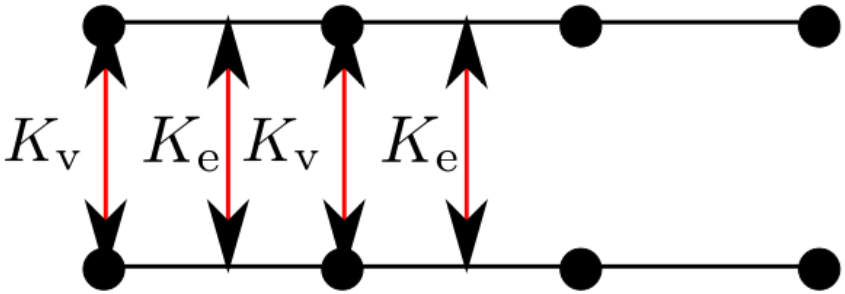
Kernel's construction framework



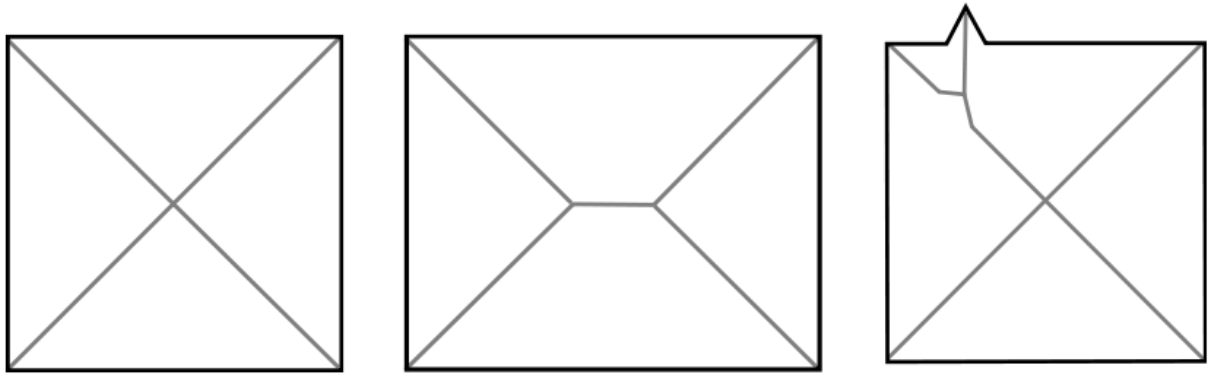
Path kernel: Usual Approach

Align paths [Kashima et al, 2002].

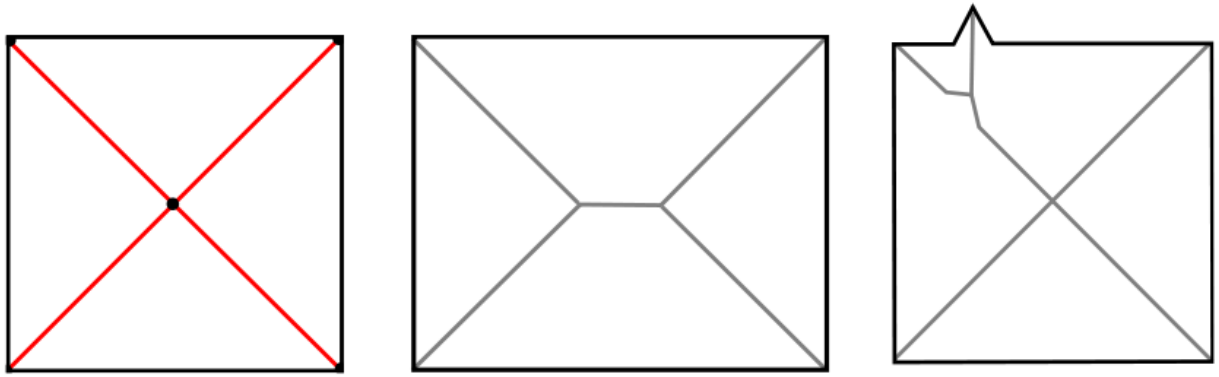
$$K(h,h') = \begin{cases} 0 & \text{if } |h| \neq [h'| \text{ and} \\ K_v(v_1,v_1').\Pi_{i=1}^{|h|} K_e(e_i,e_i')K_v(v_{i+1},v_{i+1}') & \text{otherwize} \end{cases}$$



Path kernel: Robustness against noise

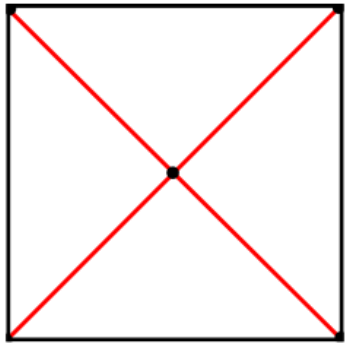


Path kernel: Robustness against noise

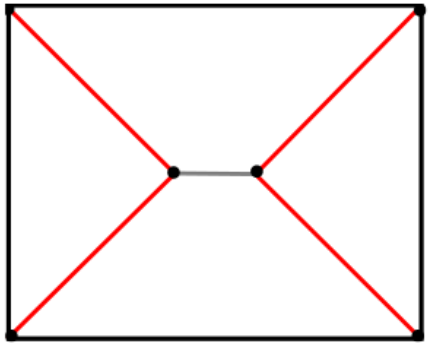


6 paths of size 2

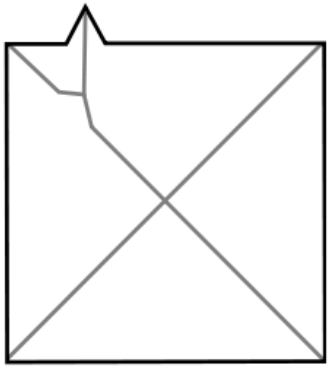
Path kernel: Robustness against noise



6 paths of size 2

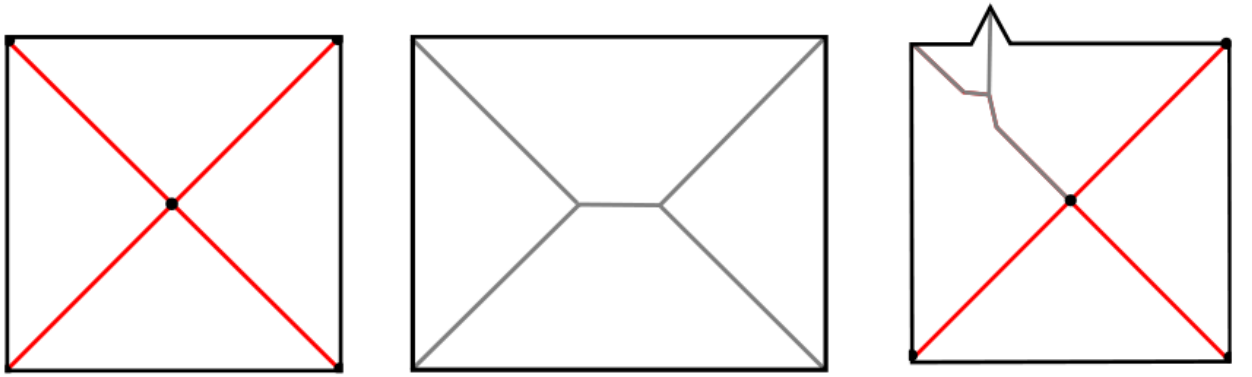


2 matched paths



Insertion
d'un arc

Path kernel: Robustness against noise



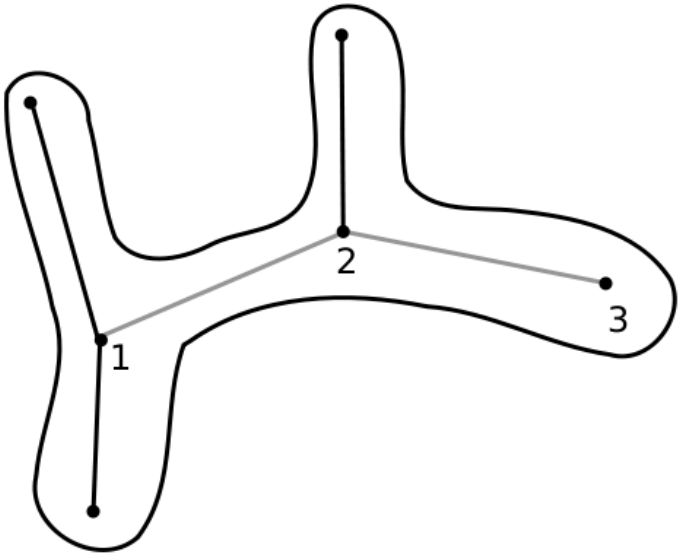
6 paths of size 2 | 2 matched paths | 3 matched paths

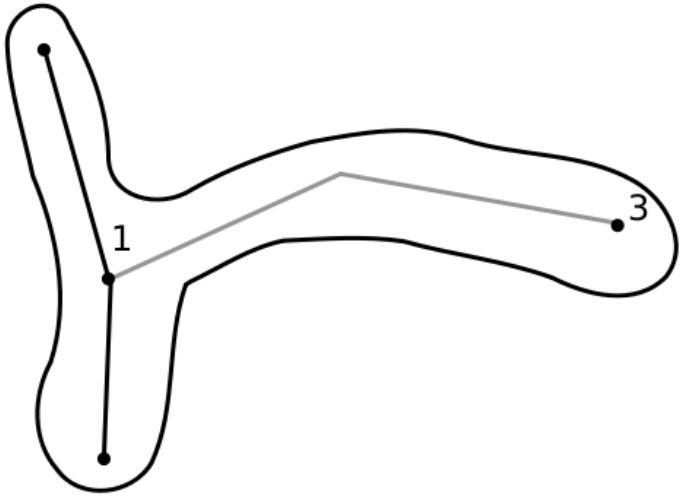
Insertion d'un noeud

Few comparable paths between similar shapes.

Comment considérer que ces formes se ressemblent malgré tout ?

Path kernel: Edit operations

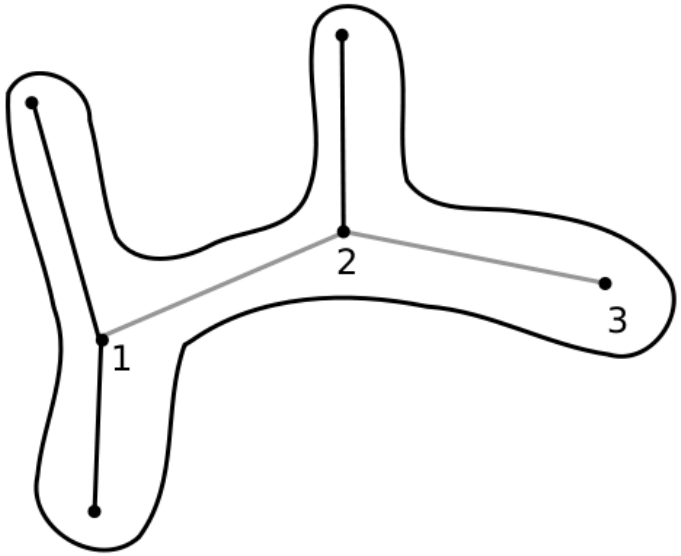


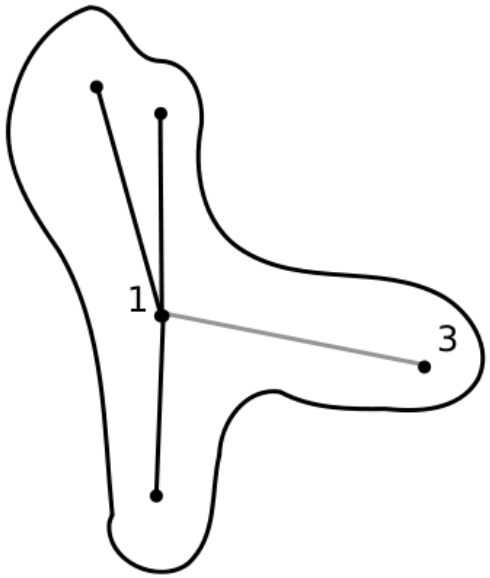


Removal of vertex 2.

'ertex Removal $\Rightarrow$ Removal of the sub-shape encoded by this vertex

Path kernel: Edit operations





Contraction of edge $e_{1,2}$.

Vertex Removal $\Rightarrow$ Removal of the sub-shape encoded by this vertex [Grady, 2005];

Edge contraction $\Rightarrow$ Removal of the sub-shape encoded by the edge.

$h \rightarrow \zeta^l(h)$ associated with a cumulative cost proportional to the removed parts of the shapes..

Noyau sur chemins édités

Comparer deux chemins $\iff$ Comparer les réécritures

$$K_{\text{edit}}(t,t') = \sum_{k=0}^{D}\sum_{l=0}^{D} \exp\left(-\frac{\text{cout}_k(t)+\text{cout}_l(t')}{2\sigma^2_{\text{cout}}}\right) K_{\text{classique}}(\zeta^k(t),\zeta^l(t')).$$

- Robuste au bruit structurel 😃 ;
- Contrôle du coût d'édition 😃 ;
- Compare des chemins de longueurs différentes 😃.

Results : Classification of 99 shapes

Classification with QDA within the Hilbert space associated to the kernel..

Noyau	**Percentage of correctly classified shapes.**
$K_{\text{edit,cover}}$	95%
$K_{\text{edit,5\%}}$	92%
$K_{\text{edit,10\%}}$	93%
$K_{\text{classic,cover}}$	91%
Random walks	82%
K_{Neuhaus}	86%

Conclusion

- Les noyaux sur graphe fournissent un plongement implicite des graphes.
- De nombreux outils statistiques peuvent être utilisés en utilisant le « kernel trick ».
- L'interprétation (cad le sens) des opérations réalisées dans l'espace de Hilbert implicite est contrôlée par le critère de similarité défini par le noyau.
- Noyaux sur graphes et analyses de formes :
    - Selection de petits sacs de chemins représentatifs
    - Construction de plusieurs noyaux sur sac de chemins.
    - Un noyau entre les chemins prend en compte la variabilité des chemins.

References

Indefinite Kernels Feature Space Interpretation of SVM with Indefinite Kernels, Bernard Haasdonk, PAMI 27(4) 2005

Kernels and Graph edit distance Bridgining the Gap Between Graph Edit Distance and kernel machines, Michel Neuhauss and Horst Bunke (Machine perception, AI, vol. 68).

Convolution Kernel Convolution Kernels on Discrete Structures TR: UCSC-CRL-99-10, David Haussler

Diffusion kernels Diffusion Kernels on graphs and Other Discrete Structures, Risi Imre Kondor, John Lafferty. ICML 2002

Kernels and infinite Bags

- Graph kernels, Journal of Machine Learning Research 11(2010) 1201-1242, S.V.N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, Karsten M. Borgwadt
- Marginalized Kernels Between Labeled Graphs, H. Kashima, K. Tsuda and A. Inokuchi, ICML-2003
- Graph kernels based on tree patterns for molecules, Pierre Mahé and Jean-Philippe Vert, Machine Learning 75(1) 3-35

Kernels and finite Bags

- Dupe, F. -X. & Brun, L.. Edition within a graph kernel framework for shape recognition. In Graph Based Representation in Pattern Recognition 2009 , pages 11-21 2009 .