Interprétation d'images II

Reconnaissance structurelle de formes/objets

Luc Brun

Scènes et objets

- Description d’une scène par :
  - des objets
    - points,
    - segments,
    - régions
  - des relations entre ces objets
    - plus proches voisins de points,. . .,
    - intersections de segments. . .,
    - adjacences (resp. k-adjacence) de régions. . .
- Buts :
  - associer une scène à un modèle,
  - une partie d’une scène à un modèle,
  - classifier des scènes par leurs contenu

$\vdots$

De la scène au grahe

- G=(V,E)
  - V ensemble des sommets : ensemble des objets,
  - E ensemble des arêtes : ensemble des relations entre ces arêtes.
- $G = (V, E)$ un graphe décrivant une scène
  - issue d'une segmentation,
  - extraction de lignes, de points…

  de l'image.
- $G' = (V', E')$ un graphe décrivant un modèle
  - objet moyen d'une classe,
  - objet théorique parfait ou connu (précédente acquisition).
- Questions :
  - G, G' peuvent ils décrire le même objet ?
  - G décrit il une partie de G' ?

Classification en termes de graphes (1/5)

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

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

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

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

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

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

Classification en termes de graphes (2/5)

- 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 (3/5)

- Soient X (image) et Y (modèle), on veut savoir si, il existe Z tel que :

$$Z \subseteqq X \text{ et } Z \subseteqq Y$$

- 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 (4/5)

- 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) comun de G et G' à un cardinal (nombre de noeuds) inférieur.

Classification en termes de graphes (5/5)

- L'appariement non bijectif :
  - Trouver $\phi : V' \to \mathcal{P}(V)$ 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.

Généralisations (1/2)

- Distances de graphes par édition (ged)
  - On défini un ensemble d'opérations permettant de passer de G à G' (ou vice versa).
    - insertion de sommet et d'arêtes,
    - substitution de label de sommets et d'arêtes,
    - suppression de sommets et d'arêtes.
  - Chaque opération est valuée par un cout.
  - le ged est la suite d'opérations de coût minimal.

Distance $\rightarrow$ métrique $\rightarrow$ (médian, clustering…)

Généralisations (2/2)

- Distance de graphes par édition étendue
  - On ajoute les deux opérations :
    - Découpe(v)$\rightarrow (v_1, \ldots, v_n)$
    - Fusionne$(v_1, \ldots, v_n) \rightarrow v$

Permet de traiter les images sur/sous segmentées.

Isomorphismes de Graphes et 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),

⋮

Approches Algorithmiques : Bunke/Ullman

- 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 $E_s = E \cap V_s \times V_s$).

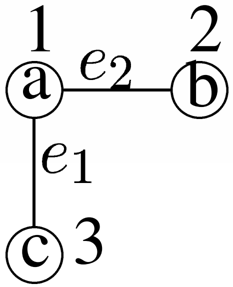
Bunke/Ullman : Definitions

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

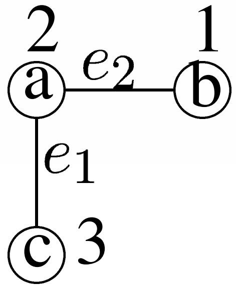
$$m_{i,j} = \begin{cases} \nu((v_i, v_j)) & \text{si } (v_i, v_j) \in E \\ 0 & \text{sinon} \end{cases}$$

- 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. $\forall i \in \{1, \ldots, n\}\ \sum_{j=0}^{n} p_{i,j} = 1$.

Matrices d'adjacences, permutations : Exemple



ou



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

$$M' = PMP^t = \begin{array}{c c} & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} b & e_2 & e_1 \\ e_2 & a & 0 \\ e_1 & 0 & c \end{pmatrix} \end{array}$$

Bunke/Ullman : Definitions

- 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.

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

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

Bunke/Ullman : Isomorphismes de sous graphes

- 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 :

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

Ullman : L'algorithme (1976)

- 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

Ullman : Backtrack

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

Ullman : Discussion

- Permet de tester les isomorphismes de sous graphes,
- Si $m = n$, on teste les isomorphismes de graphes,
- Le choix de i peut être contraint, par exemple en s'assurant que :

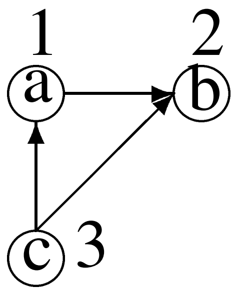
$$\forall j \in \{1, \dots, k\} p_{j,i} = 0$$

Extension par Bunke

- Comparer un graphe G à une base de donnée de modèles $\{G_1, \ldots, G_n\}$.
- Problème de Ullman : n tests d'isomorphismes !
- Solution : faire des pré calculs (ne marche que pour des graphes de tailles réduites).

Bunke

- 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

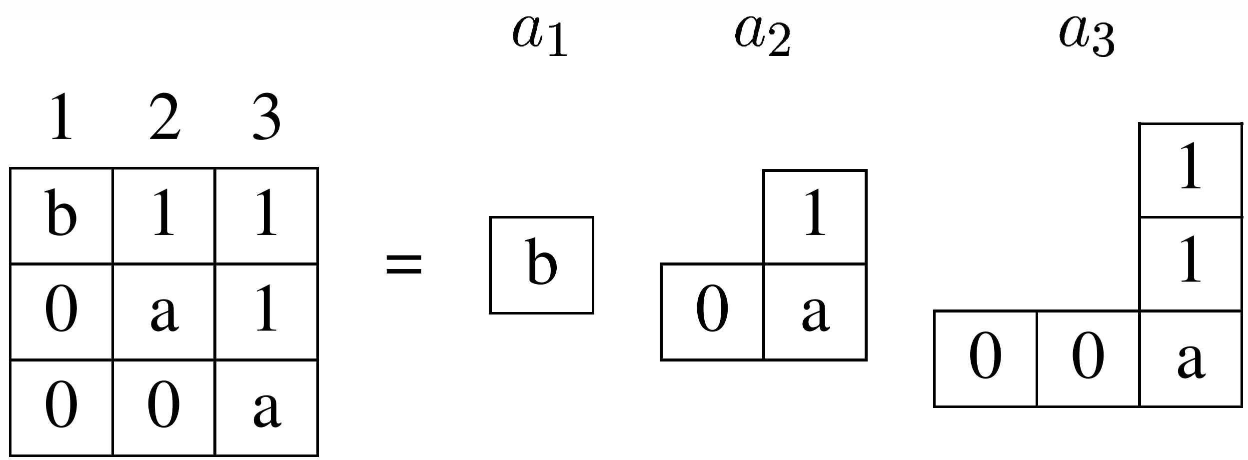
F

Bunke

- Avantage :
  - Tester si G est isomorphe à G_i reviens à tester si $M \in A(G_i)$.
- Inconvénient :
  - On est toujours linéaire en nombre de modèles,
  - Problème de l'isomorphisme de sous graphe.

Bunke

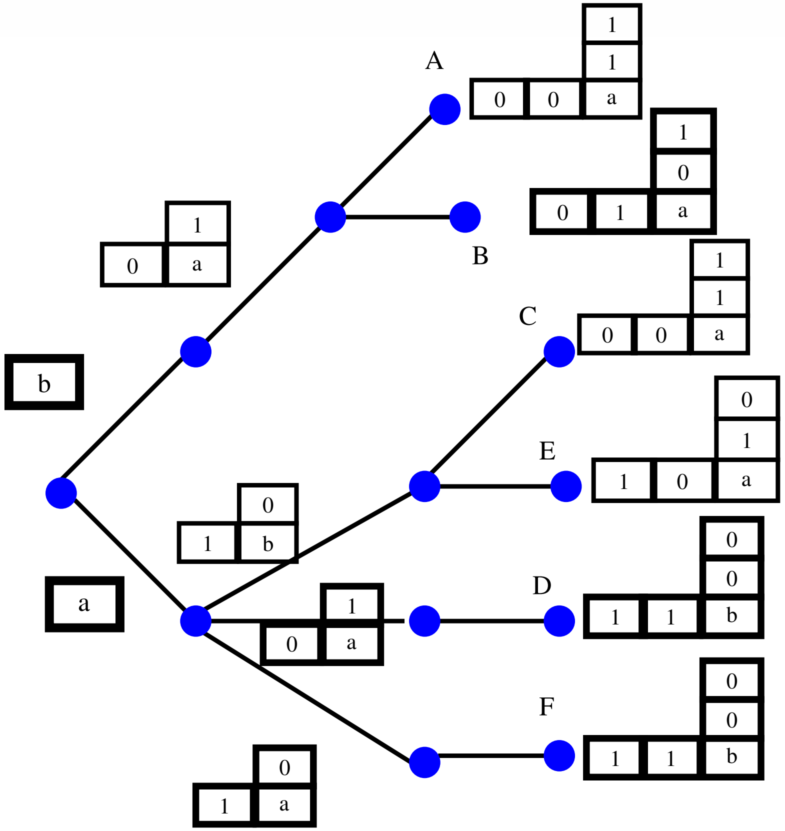
- Décomposer une matrice en vecteurs :



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

Bunke

- 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

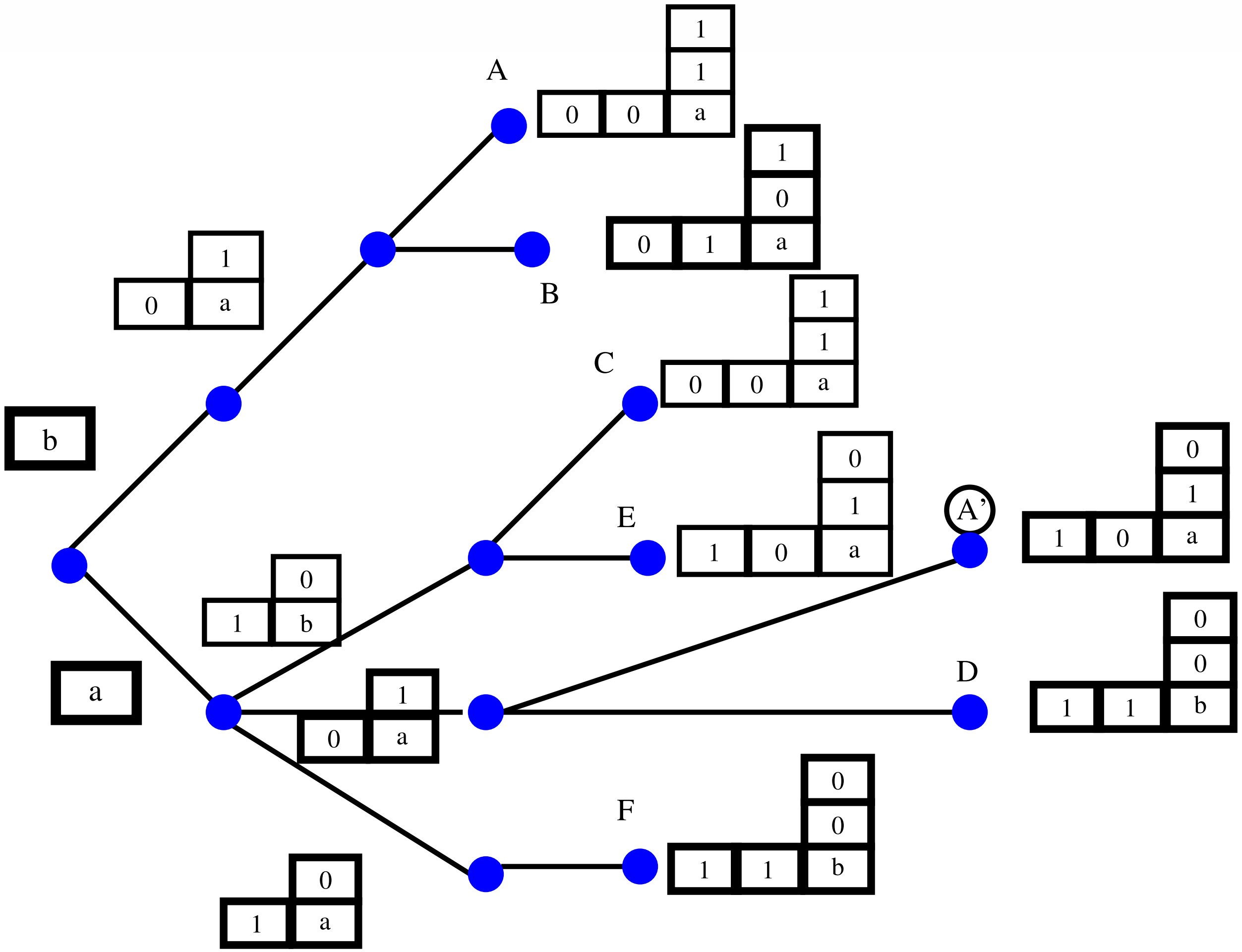
F

Bunke

- Créer un arbre contenant tous les modèles.

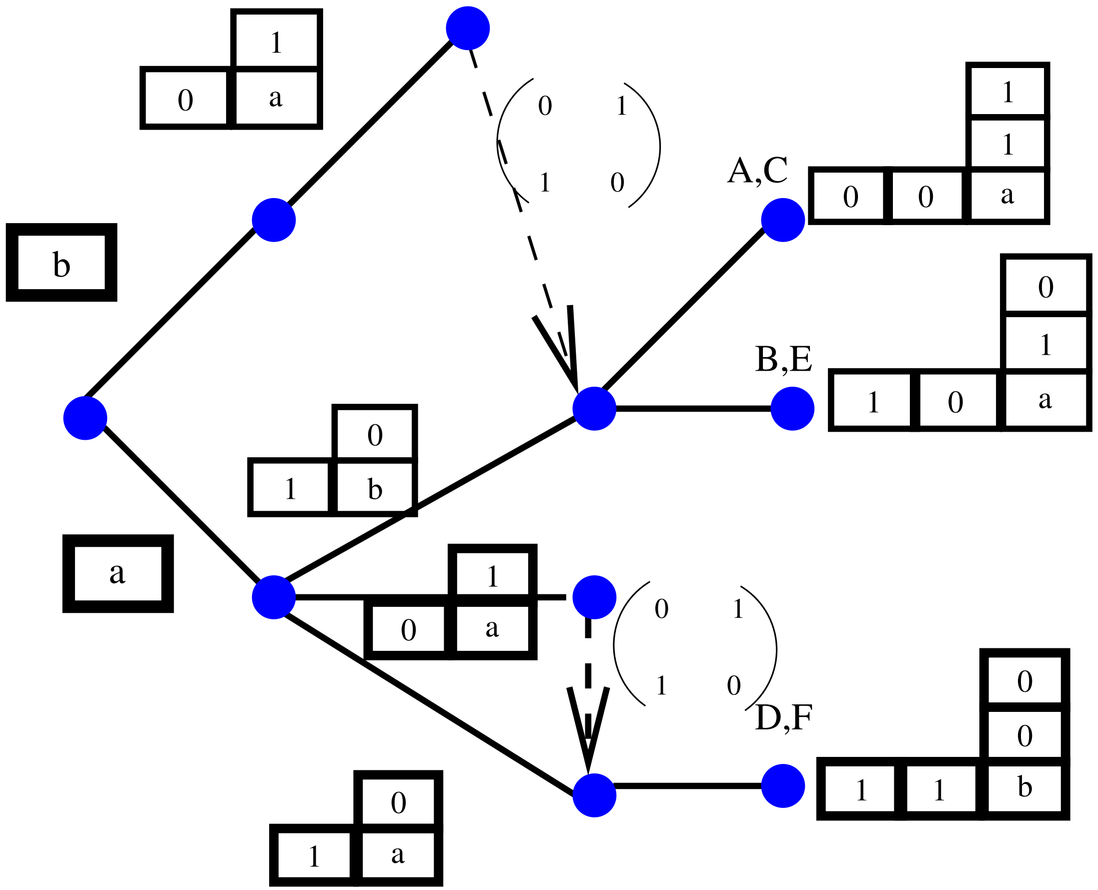
1	2	3
a	1	0
0	a	1
1	0	a

A'



Bunke

- Factoriser les branches en utilisant des permutations partielles sur les sommets.



Bunke : conclusion

- Avantages :
  - Décision extrêmement rapide,
- Inconvénients
  - Le nombre d'isomorphismes d'un graphe de taille n est égal à $n!$ (taille de A(G)).
    - Taille de l'arbre :

$$\mathcal{O}(|L_v|\left(1+|L_e|^2\right)^M) \text{ avec } M = max_i|V_i|$$

    - Temps de recherche

$$\mathcal{O}(M^2|L_e| + M|L_v|)$$

- Conclusion
  - Méthode réservé à des bases de données de graphes de petites tailles.

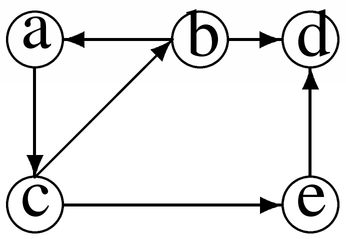
Appariement par représentation de l'espace des

- 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.

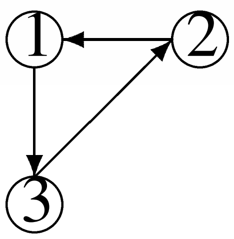
SSR : Algorithme de Cordella 2001 (VF2)

- 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 $M_2(s)$.

VF2 : Exemple de notations



G_1



G_2

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

VF2 : 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

VF2 : 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.

VF2 : Calcul de $F(s, n, m)$

$$\begin{aligned} F(s,n,m) &= R_{pred}(s,n,m) \wedge R_{succ}(s,n,m) \wedge \\ & \quad R_{in}(s,n,m) \wedge R_{out}(s,n,m) \wedge R_{new}(s,n,m) \end{aligned}$$

- 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} : Pourrais je arriver à un apariement (a terme) ?
  - $R_{pred}(s, m, n)$: les prédécesseurs correspondent :

$$\begin{aligned} & (\forall n' \in M_1(s) \cap Pred(G_1, n) \exists m' \in Pred(G_2, m) | (n', m') \in M(s)) \wedge \\ & (\forall m' \in M_2(s) \cap Pred(G_2, m) \exists n' \in Pred(G_1, n) | (n', m') \in M(s)) \end{aligned}$$

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

$$\begin{aligned} & (\forall n' \in M_1(s) \cap Succ(G_1, n) \exists m' \in Succ(G_2, m) | (n', m') \in M(s)) \wedge \\ & (\forall m' \in M_2(s) \cap Succ(G_2, m) \exists n' \in Succ(G_1, n) | (n', m') \in M(s)) \end{aligned}$$

Prédicats R_{in} et R_{out}

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

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

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

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

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

Prédicat R_{new}

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

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

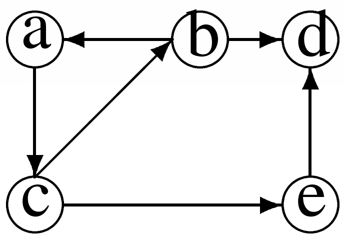
  - avec :
    - $N_1(s) = V_1 - M_1(s) - T_1^{in}(s) \cup T_1^{out}(s)$: tout ce qui reste a voir dans G_1.
    - $N_2(s) = V_2 - M_2(s) - T_2^{in}(s) \cup T_2^{out}(s)$: tout ce qui reste a voir dans G_2.

VF2 : Exemple (1/)

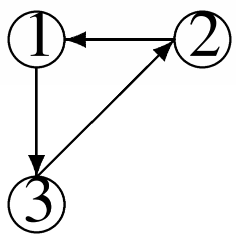
G_1 G_2

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

VF2 : Exemple



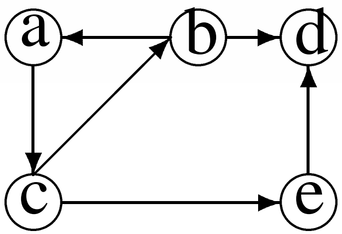
G_1



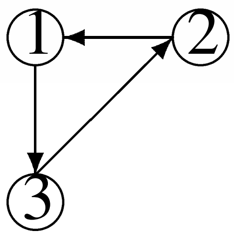
G_2

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

VF2 : 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.

VF2 : 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).

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 $\leftarrow$ taille(M(s'))
MCS $\leftarrow$ M(s')
finsi
appariement(s')
finsi

Graphe d'association

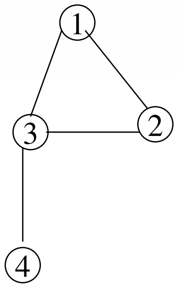
- Soient $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$, le graphe d'association $G = (V, E)$ de G_1 et G_2 est défini par :
  - Sommets :

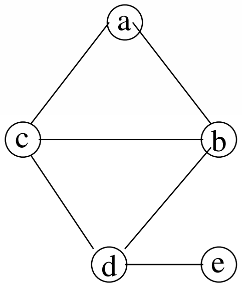
$$V = V_1 \times V_2$$

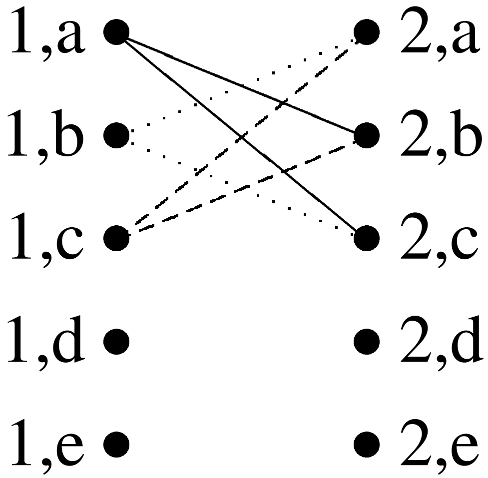
  - Arêtes :

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

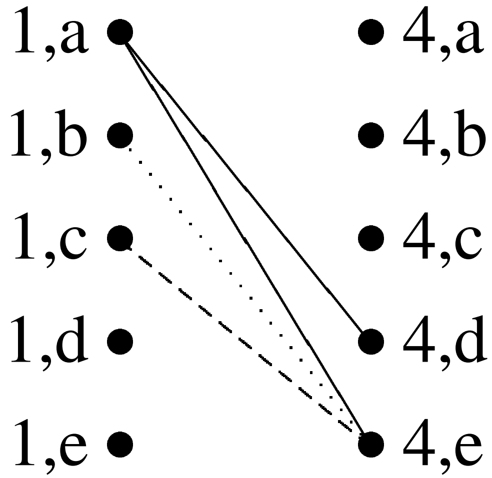
Graphe d’association : Exemple



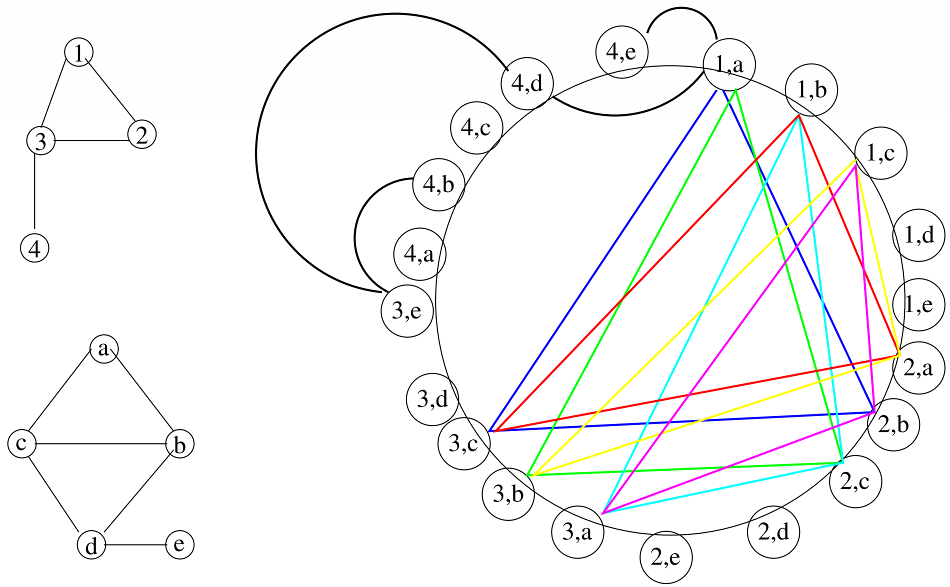




mais également



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, $\omega(G)$ est la taille (en nombre de sommets) du clique maximum.
- Théorème :

  Soient G_1 et G_2 deux graphes et G leurs graphe d'association. Il existe une relation bijective entre
  - *les cliques maximal (maximum) de G et*
  - *les sous graphes communs maximal (maximum) de G_1 et G_2.*
- Donc : Calculer les cliques du graphe d'association G est **équivalent** à calculer les sous graphes communs de G_1 et G_2.

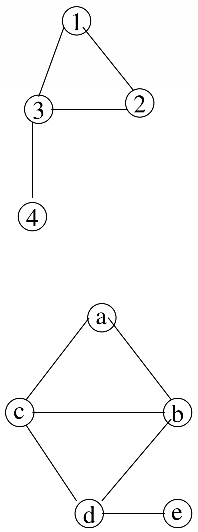
L'algorithme de DurandParisi (1999)

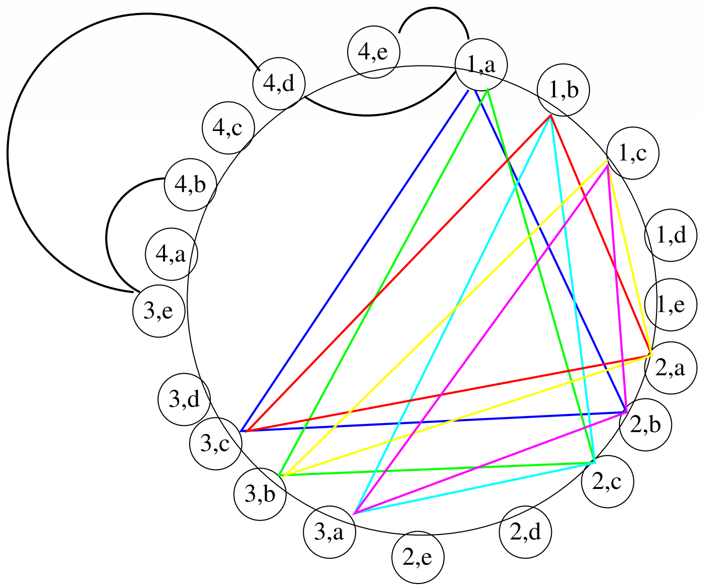
```
procédure durandParisi (E s)
début
    tant que nextNode(s,n) faire
        si isLegalNode(s,n) et non pruningCondition(s) alors
            s' ← addNode(s,n)
            si taille(s')> tailleMax alors
                MCS ← M(s')
                tailleMax ← taille
            finsi
            si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

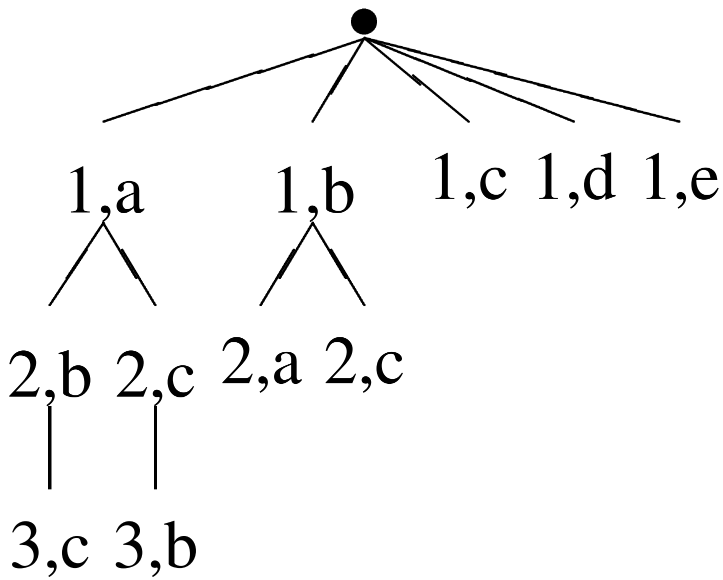
DurandParisi : Remarques

- Au niveau k on considère les sommets $n = (k, n_2)$ de façon à garantir que l'on ne bouclera pas.
- Si un sommet $k \in G_1$ ne peut pas être apparié dans G_2, on ajoute le sommet spécial $\emptyset$ et on continu la recherche au niveau $k + 1$.

DurandParisi : Exemple d’exécution







DurandParisi : Conclusion

- Très similaire au VF2.
- Différence sur l'utilisation du clique.
  - ☺ On ne calcule presque rien,
  - ☹ On stocke énormément de choses (le graphe d'association).

Les algorithmes $SM^i, i = 1, 2$

- Basés sur un pré-calcul de tous les cliques de taille i.
  - $i = 1$: sommets,
  - $i = 2$ couples de sommets.
- Notations :
  - Voisinage :

$$N_j = \{k \in V \,|\, (j,k) \in E\}$$

  - Candidats à l'adjonction au clique K :

$$C_0(K) = \{j \in V - K \,|\, \forall k \in K \; (j,k) \in E\} = \bigcap_{k \in N_k} N_k$$

SM^i : L'algorithme

procédure SM^i **(E G)**

Déclaration Q : cliques de cardinal i,

$\mathcal{K}$,K^*,max

début

$\mathcal{K}\leftarrow\emptyset$;$K^*\leftarrow\emptyset$;max$\leftarrow 0$

tant que Q $\neq \emptyset$ **faire**

H $\leftarrow$ pop(Q)

K $\leftarrow$ H

tant que $C_0(K) \neq \emptyset$ **faire**

l $\leftarrow$ $arg\max_{j \in C_0(K)} |C_0(K) \cap N_j|$

K $\leftarrow K \cup \{l\}$

fintantque

$\mathcal{K} \leftarrow \mathcal{K} \cup K$

si $|K|$>max **alors**

max $\leftarrow |K|$

$K^* \leftarrow$ K

finsi

fintantque

fin

SM^i : Commentaires

- Essaye de construire autant de cliques maximal que de cliques initiaux de taille i.
- Converge pour chaque clique initial vers un optimum local.
- SM^2 plus efficace mais plus lent que SM^1.
- SM^1_SWAP compromis entre SM^1 et SM^2
  - But : Essayer d'explorer autour des optimums locaux.

SM^1_SWAP : Notations

- Candidats à un échange :

$$C_1(K) = \{j \in V - K \,|\, |N_j \cap K = |K| - 1\}$$

- $j \in C_1(K)$ ssi un seul sommet de K lui interdi d'entrer dans le clique.
- Soient $l \in C_1(K)$ et $k_l \in K$ tel que $(l, k_l) \notin E$. Si on ajoute l à K, il faut enlever k_l.

$$K = K \cup \{l\} - \{k_l\}$$

SM^1_SWAP : Choix du sommet à ajouter

$$l = arg \max_{j \in C_0(K)} |C_0(K) \cap N_j| \text{ ou } l = arg \max_{j \in C_0(K) \cup C_1(K)} |C_0(K) \cap N_j| \text{ ?}$$

- Le choix d'un sommet dans $C_1(K)$ permet de s'écarter de l'optimum local mais :
  - ce choix n'améliore pas nécessairement l'optimum,
  - ne nous rapproche pas nécessairement de la convergence,
  - peut nous amener à boucler.
- On se restreint à $C_0(K)$:
  - durant un nombre fixe d'itérations ($START\_SWAP$),
  - Lorsque le nombre d'échanges est supérieur à un multiple T de $|K|$,
  - lorsque le sommet sélectionné est celui supprimé par le dernier échange.

SM^1_SWAP : algorithme de choix

fonction select ($G, K, last\_swap$) **:** Sommet

Déclaration l : Sommet

début

si $n\_swap \leq T|K|$ et $|K| \geq START\_SWAP$ **alors**

$l \leftarrow arg\max_{j \in C_0(K) \cup C_1(K)} |C_0(K) \cap N_j|$

si $l = last\_swap$ **alors**

$l \leftarrow arg\max_{j \in C_0(K)} |C_0(K) \cap N_j|$

finsi

sinon

$l \leftarrow arg\max_{j \in C_0(K)} |C_0(K) \cap N_j|$

finsi

retourner l

fin

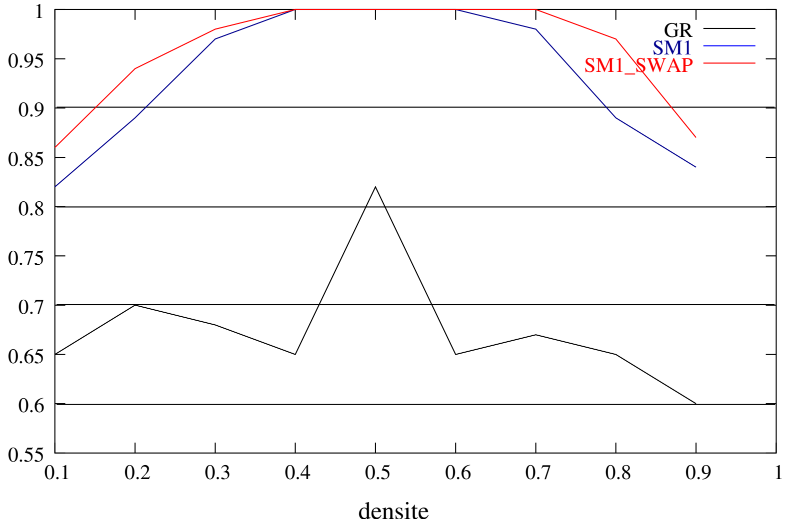
- On suppose que $arg\max_{j \in C}$ renvoi $\emptyset$ si C est vide.

SM^1_SWAP : L'algorithme

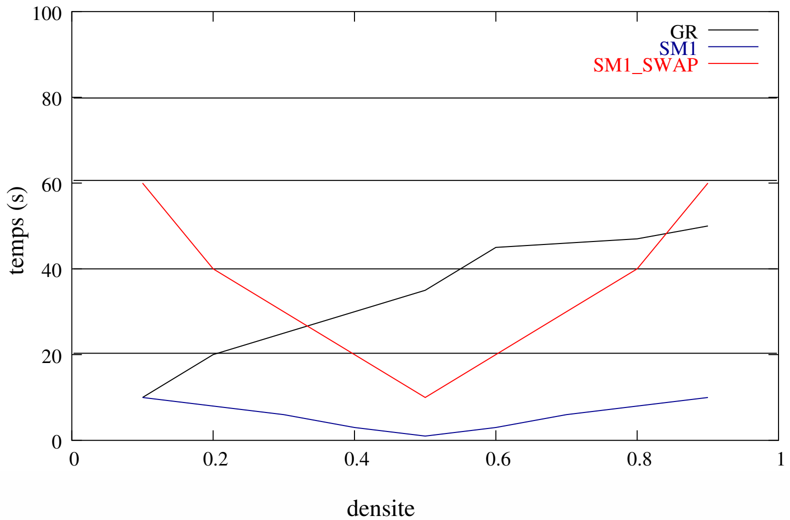
procédure SM^1_SWAP **(E** $G = (V, E)$ **)**
  Déclaration $\mathcal{K}$,K^*,W,max,W
début
  $\mathcal{K}\leftarrow\emptyset$;$K^*\leftarrow\emptyset$;max$\leftarrow$0 ;$W\leftarrow V$
  tant que W $\neq \emptyset$ **faire**
    h $\leftarrow$ pop(W)
    K $\leftarrow \{h\}$
    $n\_swap\leftarrow 0$;$last\_swap\leftarrow\emptyset$
    l $\leftarrow$ select($G, K, last\_swap$)
    tant que $l \neq \emptyset$ **faire**
      si $l \in C_0(K)$ **alors**
        $K \leftarrow K \cup \{l\}$
      sinon
        $n\_swap \leftarrow n\_swap + 1$
        $last\_swap \leftarrow k_l$
        $K \leftarrow K \cup \{l\} - \{k_l\}$
        push(W,k_l)
      finsi
      l $\leftarrow$ select($G, K, last\_swap$)
    fintantque
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{K\}$
    si $|K|$>max **alors**
      max $\leftarrow |K|$
      $K^* \leftarrow$ K
    finsi
  fintantque
fin

SM^1_SWAP : Performances

- taille clique/ taille max : $\frac{|K|}{\omega(G)}$



- Temps de calculs en secondes :



Méthode par 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

- On considère $G = (V, E)$ et $C \subset V$ et $|V \models n$
- Vecteur caractéristique de C :

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

- Tout vecteur caractéristique appartient au simplex de dim n :

$$\forall C \subset V \; x^C \in S_n = \{x \in \mathbf{R}^n \,|\, e^t x = 1 \text{ et } \forall i \in \{1, \ldots, n\} x_i \geq 0\}$$

Fonction quadratique

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

$$a_{i,j} = \begin{cases} 1 & \text{si } (i,j) \in E, \\ 0 & \text{sinon.} \end{cases}$$

- et la fonction :

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

- x^* est un maxima strict de g ssi :

$$\exists \epsilon > 0 \mid \begin{cases} |y - x^*| < \epsilon & \Rightarrow \quad f(y) \leq f(x^*) \\ |y - x^*| < \epsilon \text{ et } f(y) = f(x^*) & \Rightarrow \quad y = x^* \end{cases}$$

Théorème de Motzkin-Straus(65)-Bomze(97)

- Théorème : Soit $S \subset V$ et x^S son vecteur caractéristique alors :
  1. S est un clique maximum de G ssi x^S est un maximum global de g sur S_n. On a alors :
  $$\omega(G) = \frac{1}{2(1 - g(x^S))}$$
  2. S est un clique maximal de G ssi x^S est un maximum local de g sur S_n.
  3. Tout maxima local (et donc global) x de g sur S_n est strict et de la forme $x = x^S$ avec $S \subset V$.

Application à la reconnaissance d'objets

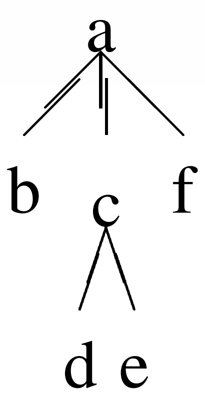
- Représenter chaque objet par un schock tree $A = (V, E)$
  - Un arbre enraciné représentant le squelette
  - Feuilles : détails de l'objet
- Chemin entre deux sommets unique :

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

- Chaine entre sommets :

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

Graphe d'association

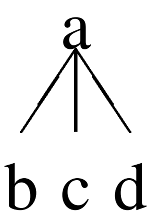


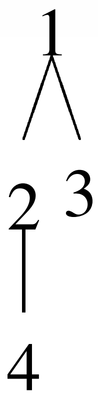
$$str(b,e) = 1. - 1. - 1$$

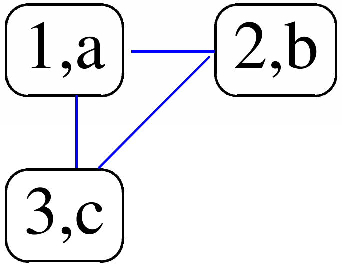
- Soient $A_1 = (V_1, E_1)$ et $A_2 = (V_2, E_2)$.
- On construit le graphe d'association $G = (V_1 \times V_2, E)$ avec :

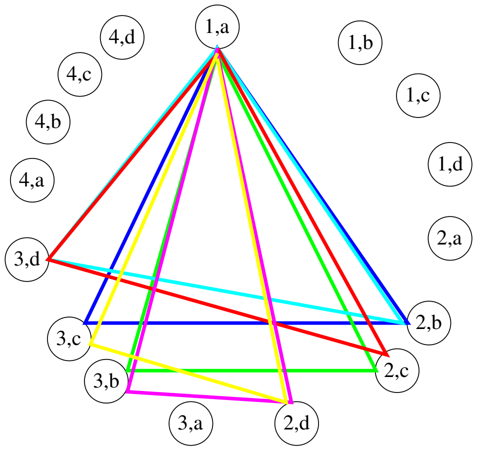
$$((u,w),(v,z)) \in E \Leftrightarrow Str(u,v) = Str(w,z)$$

Graphe d’association : Exemple









Résolution

- Recherche des maxima de

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

- 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($x_i(t+1) = x_i(t)$).

Gibbons, Bomze (97)

- Soit la matrice $A_{\overline{G}} = (\overline{a}_{ij})$ avec $\overline{a}_{i,j} = 1$ si $(i,j) \notin E$, 0 sinon.

$$A_{\overline{G}} = ee^t - A_G - I$$

- Transformation du problème :

$$\begin{aligned}
f(x) &= x^t \overline{A} x = x^t \left(A_{\overline{G}} + \tfrac{1}{2} I \right) x \\
&= x^t \left(ee^t - A_G - I + \tfrac{1}{2} I \right) x \\
&= x^t \left[ee^t - \left(A_G + \tfrac{1}{2} I \right) \right] x \\
&= x^t ee^t x - x^t \left(A_G + \tfrac{1}{2} I \right) x \\
&= 1 - g(x)
\end{aligned}$$

Théorème de Gibbons, Bomze(97)

- Théorème : Soit $S \subset V$ et x^S son vecteur caractéristique alors :
  1. S est un clique maximum de G ssi x^S est un minimum global de f sur S_n. On a alors :
  $$\omega(G) = \frac{1}{2f(x^*)}$$
  2. S est un clique maximal de G ssi x^S est un minimum local de f sur S_n.
  3. Tout minimum local (et donc global) x de g sur S_n est strict et de la forme $x = x^S$ avec $S \subset V$.

Intuition du théorême (1/2)

- Formulation du problème

$$\begin{aligned} f(x) &= x^t \overline{A} x = x^t (A_{\overline{G}} + \tfrac{1}{2} I) x \\ &= \textstyle\sum_{i=1}^n \frac{1}{2} x_i^2 + \sum_{j|(i,j)\notin E} x_i x_j \end{aligned}$$

- Pour x^C vecteur caractéristique de $C \subset V$:

$$\begin{aligned} f(x^C) &= \tfrac{|C|}{2|C|^2} + \textstyle\sum_{i=1}^n \sum_{j|(i,j)\notin E} x_i^C x_j^C \\ &= \tfrac{1}{2|C|} + \textstyle\sum\sum_{(i,j)\in C^2|(i,j)\notin E} x_i^C x_j^C \end{aligned}$$

Si C est un clique :

$$f(x) = \frac{1}{2|C|}$$

- Plus C est «grand» plus, $f(x)$ est «petit».

Intuition du théorême (2/2)

- Supposons que l'on ajoute à C un sommet qui est adjacents à tous les sommets de C sauf 1.
- Soit $x^{C'}$ le vecteur résultant :

$$\begin{aligned} f(x^{C'}) &= \frac{1}{2|C|} + \sum\sum_{(i,j)\in C'^2 \mid (i,j)\notin E} x_i^{C'} x_j^{C'} \\ &= \frac{1}{2(|C|+1)} + \frac{1}{(|C|+1)^2} \end{aligned}$$

- $f(x^{C'})$ est il plus grand ou plus petit que $f(x^C)$?

$$\begin{aligned} \frac{1}{2(|C|+1)} + \frac{1}{(|C|+1)^2} &> \frac{1}{2|C|} \\ &\Updownarrow \\ |C|+1+2 &> \frac{(|C|+1)^2}{|C|} = |C|+2+\frac{1}{|C|} \\ &\Updownarrow \\ 3 &> 2+\frac{1}{|C|} \end{aligned}$$

Intérêt de la reformulation

- Application aux graphes valués.
- Clique maximum :

$$\omega(G) = max\{|S| \text{ tel que } S \text{ est un clique de } G\}$$

- Clique maximum pondéré.
  - Soit un vecteur de poids : $w \in \mathbf{R}^n$

$$\omega(G, w) = max\{W(S) \text{ tel que } S \text{ est un clique de } G\}$$

avec

$$W(S) = \sum_{i \in S} w_i$$

Cliques pondérés

Les cliques de poids maximal (resp. maximum) correspondent aux minimums locaux (resp. global) de :

$$f(x) = x^t C(w) x$$

avec

$$C(w)_{i,j} = \begin{cases} \frac{1}{2w_i} & \text{si } i = j \\ \frac{1}{2w_i} + \frac{1}{2w_j} & \text{si } i \neq j \text{ et } (i,j) \notin E \\ 0 & \text{sinon} \end{cases}$$

- Possibilité de fournir des informations a priori !
  - distance entre points,
  - similarité de régions…

Résolution

- Le problème : minimiser sur S_n

$$f(x) = x^t A x$$

- Formulation en terme de «Linear Complementarity Problem» (LCP)
  - Trouver y, $\overline{x}$ tels que : $y = q_G + M_G \overline{x} \geq 0$, $\overline{x} = [x, x_{n+1}, x_{n+2}]$,

$$q_G = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ 1 \end{pmatrix} \text{ et } M_G = \begin{pmatrix} A & -e & e \\ e^t & 0 & 0 \\ -e^t & 0 & 0 \end{pmatrix}$$

Résolution : Remarques

- LCP :Méthode itérative basée sur le choix d'un pivot à chaque étape.
- Locatelli& Pellilo ont proposé une heuristique de choix des pivots $\rightarrow$ algorithme PBH.
- Algorithme PBH formellement équivalent à l'algorithme SM^1.

Utilisation de méthodes de SoftAssign

- On considère deux graphes G et g avec des arêtes valuées.
  - $G_{a,b}$ poids de l'arête (a,b) dans G
  - $g_{i,j}$ poids de l'arête (i,j) dans g
- On considère :
  - Une matrice de permutation M ($M_{a,i} = 1$) si a est associé à i, 0 sinon.
  - Une fonction de distance entre les attributs des arêtes :

$$C_{a,b,i,j} = \begin{cases} 0 & \text{Si } G_{a,b} \text{ ou } g_{i,j} \text{ est null} \\ c(G_{a,b}, g_{i,j}) & \text{sinon} \end{cases}$$

avec (par exemple) :

$$c(G_{a,b}, g_{i,j}) = 1 - 3|G_{a,b} - g_{i,j}|$$

SoftAssign : Le problème

- On voudrait minimiser :

$$Ewg(M) = -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{a,i}M_{b,j}C_{a,i,b,j}$$

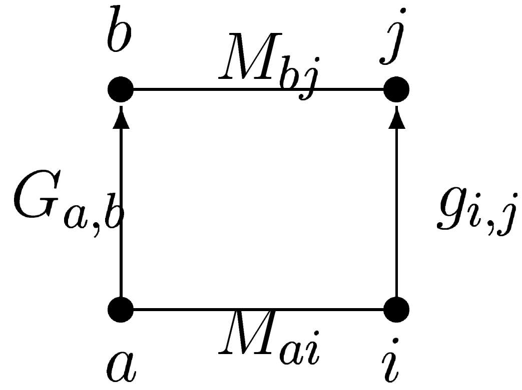
avec A (resp. I) nb sommets dans G (resp. g).

- i.e. apparier un maximum d'arêtes qui se ressemblent.
- Si $G_{a,b}, g_{i,j} \in \{1, NULL\}$, $C_{a,b,i,j} \in \{0, 1\}$ et

$$E_{wg}(M) = -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{a,i}M_{b,j}G_{a,b}g_{i,j}$$

SoftAssign : Illustration

- Ce que l'on cherche.



SoftAssign : Introduction

- Soit le problème restreint : Étant donné $\{X_1, \ldots, X_n\}$ déterminer $\{m_1, \ldots, m_n\}$ tel que
  - $m_i = 1$ si $X_i = max_j X_j$,
  - 0 sinon
- équivalent à maximiser $\sum_{i=1}^{n} m_i X_i$
- On pose pour $\beta > 0$:

$$m_j(\beta) = \frac{e^{\beta X_j}}{\sum_{i=1}^{n} e^{\beta X_i}}; \lim_{\beta \to +\infty} m_j(\beta) = \begin{cases} 1 & \text{si } X_j = \max X_i \\ 0 & \text{sinon} \end{cases}$$

SoftAssign : Algorithme1

procédure softassign1 ($\{X_1, \ldots, X_n\}$)
    Déclaration $\{m_1, \ldots, m_n\}$
début
    $\beta \leftarrow \beta_0$
    tant que $\beta < \beta_f$ **faire**
        $m_i \leftarrow e^{\beta X_j}$
        $m_i \leftarrow \frac{m_i}{\sum_{i=1}^{n} m_i}$
        Faire le reste de l'algortihme
        Incrémenter β
    fintantque
fin

- On décide petit à petit (d'où le nom soft assign)

SoftAssign : Introduction 2

- Soit à présent une matrice de permutation M, entre deux graphes G et g et une variable $X_{a,i}$.
- Maximiser en M :

$$E_{ass}(M) = \sum_{a=1}^{A} \sum_{i=1}^{I} M_{a,i} X_{a,i}$$

- On a plus une contrainte $\left(\sum_{i=1}^{n} m_i = 1\right)$ mais deux :
  - $\forall a \in \{1, \ldots, A\} \sum_{i=1}^{I} M_{a,i} = 1$
  - $\forall i \in \{1, \ldots, I\} \sum_{a=1}^{A} M_{a,i} = 1$

SoftAssign : Algorithme 2

- On normalise itérativement sur les lignes et les colonnes $\approx$ on applique l'algorithme 1 sur les lignes puis sur les colonnes.

procédure softassign2 (X_{ai})
  Déclaration M
début
  $\beta \leftarrow \beta_0$
  tant que $\beta < \beta_f$ **faire**
    $M_{a,i} \leftarrow e^{\beta X_{ai}}$
    répéter

$$M_{a,i} \leftarrow \frac{M_{a,i}}{\sum_{j=1}^{I} M_{a,j}}$$

$$M_{a,i} \leftarrow \frac{M_{a,i}}{\sum_{x=1}^{A} M_{x,i}}$$

    jusqu'à ce que M converge
    Faire le reste
    Incrémenter β
  fintantque
fin

- M converge vers une matrice de permutation.

SoftAssign : Vers la solution (1/2)

- Problème : L'appariement n'est pas une recherche de max.

$$arg \max \sum_{a=1}^{A} \sum_{i=1}^{I} M_{a,i} X_{a,i} \neq arg \min -\frac{1}{2} \sum_{a=1}^{A} \sum_{i=1}^{I} \sum_{b=1}^{A} \sum_{j=1}^{I} M_{a,i} M_{b,j} C_{a,i,b,j}$$

- Considérons $E_{wg}(M) = -\frac{1}{2} \sum_{a=1}^{A} \sum_{i=1}^{I} \sum_{b=1}^{A} \sum_{j=1}^{I} M_{a,i} M_{b,j} C_{a,i,b,j}$ comme une fonction de AI variables.
- Appliquons Taylor à l'ordre 1 :

$$E_{wg}(M) \approx E_{wg}(M^0) + \sum_{a=1}^{A} \sum_{i=1}^{I} \frac{\partial E_{wg}(M)}{\partial M_{ai}} (M_{a,i} - M^0_{a,i})$$

SoftAssign : Vers la solution

- On a donc

$$
\begin{aligned}
E_{wg}(M) &= -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{a,i}M_{b,j}C_{a,i,b,j} \\
&\approx E_{wg}(M^0) + \sum_{a=1}^{A}\sum_{i=1}^{I}\frac{\partial E_{wg}(M^0)}{\partial M_{ai}}(M_{a,i} - M_{a,i}^0)
\end{aligned}
$$

avec :

$$\frac{\partial E_{wg}(M^0)}{\partial M_{a,i}} = -\sum_{b=1}^{A}\sum_{j=1}^{I} M_{b,j}^0 C_{a,i,b,j}$$

SoftAssign : Vers la solution

- Posons $Q_{a,i} = -\frac{\partial E_{wg}(M^0)}{\partial M_{a,i}}$
- On a :

$$\begin{aligned} E_{wg}(M) &\approx E_{wg}(M^0) - \sum_{a=1}^{A} \sum_{i=1}^{I} Q_{ai}(M_{a,i} - M^0_{a,i}) \\ &\approx Cte - \sum_{a=1}^{A} \sum_{i=1}^{I} Q_{a,i} M_{a,i} \end{aligned}$$

- Minimiser $E_{wg}(M)$ reviens à maximiser :

$$\sum_{a=1}^{A} \sum_{i=1}^{I} Q_{a,i} M_{a,i}$$

Un problème de calcul de max !

La méthode

1. Choisir une valeur initiale de M
2. Faire une décomposition de Taylor de $E_{wg}(M)$
3. Faire un softassign correspondant au calcul du max de

$$\sum_{a=1}^{A}\sum_{i=1}^{I} Q_{a,i} M_{a,i}$$

4. prendre le M résultant et boucler en incrémentant β

- Remarque : On ajoute une ligne et une colonne à M pour transformer les inégalités $\sum_{a=1}^{A} M_{a,i} \leq 1$ et $\sum_{i=1}^{I} M_{a,i} \leq 1$ en égalités$\rightarrow$ matrice $\tilde{M}$.
- Permet de coder les sommets non appariés.

L'algorithme

procédure apparier (G, g, β_f, β_0)

  Déclaration β, M

début

  $\beta \leftarrow \beta_0$

  $\tilde{M}_{a,i} \leftarrow 1 + \epsilon$

  tant que $\beta < \beta_f$ **faire**

    répéter

      $Q_{a,i} \leftarrow -\frac{\partial E_{wg}(M)}{\partial M_{a,i}}$

      $M^0_{a,i} \leftarrow e^{\beta Q_{a,i}}$

      répéter

        $\tilde{M}^1_{a,i} \leftarrow \frac{\tilde{M}^0_{a,i}}{\sum_{i=1}^{I+1} \tilde{M}^0_{a,i}}$

        $\tilde{M}^0_{a,i} \leftarrow \frac{\tilde{M}^1_{a,i}}{\sum_{a=1}^{A+1} \tilde{M}^1_{a,i}}$

      jusqu'à ce que $\tilde{M}$ converge ou nb iter $> I_1$

    jusqu'à ce que M converge ou nb iter $> I_0$

    $\beta \leftarrow \beta_r \beta$

  fintantque

  seuiller $M_{a,i}$

fin

Extension aux attributs de sommets

- Intégrer une fonction $C^{(1)}_{a,i}$ de distance entre sommets

$$
\begin{aligned}
E_{arg}(M) = \quad & -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{a,i}M_{b,j}C^{(2)}_{a,i,b,j} \\
& +\alpha\sum_{a=1}^{A}\sum_{i=1}^{I} M_{a,i}C^{(1)}_{a,i}
\end{aligned}
$$

- Reviens à ajouter $\alpha C^{(1)}_{a,i}$ à $Q_{a,i}$ dans l'algorithme précédent.