Preuve de l’algorithme

Vocabulaire

On définit S comme le graphe représentant notre structure et M comme le graphe représentant notre motif.

Un motif approximé de M est un match(a,b) tels que a soit isomorphique de M au degré de tolérance près.

Les sous-graphe d’un match(a,b) sont les sous-graphes formés par l’ensemble des nœuds de a (pour le sous-graphe de structure) ou b (pour le sous-graphe de motif).

Un match(a,b) est un ensemble d'association entre une arête de S et une arête de M.

Le degré de tolérance est le nombre maximal du nombre de différences entre les deux sous-graphs d’un match.
On distingue plusieurs différences: Type de liaison différente, Arêtes en plus, Arêtes en moins

Un match(a,b) est un ensemble d'association entre une arête a(i) appartenant à S et une arête b(i) appartenant M. Il est de taille i lorsque la taille de ses sous-ensemble est i

On agrandi un match(a,b) lorsqu’on augmente les sous-ensemble a,b en ajoutant des association de nouvelles arêtes de S avec de nouvelles arêtes de M (n’appartenant pas déjà à a et b)

Une solution(A,B) est un match(A,B) où l’ensemble des nœuds composant les arêtes de a forme un sous-graph de S considéré comme isomorphique avec notre motif en prenant en compte l’ensemble des degrés de tolérance. Alors A est un motif approximatif de M.

Un bon match(a,b) est un sous-ensemble d’association d'arêtes tel qu’il existe une solution(A,B) avec:
a soit un sous-ensemble de A
b soit un sous-ensemble de B
les match de a(i) et b(i) sont identiques à ceux de A(i) et B(i)

Un mauvais match(a,b) est un sous-ensemble d'association n’était un bon match d’aucune solution.

Preuve

<u>Initialisation de l'algorithme:</u>

Une arête de M est choisie par l'algorithme (arête de départ) , pour chaque arête de S on l'associe avec l'arête de départ dans un nouveau match.

On a forcément dans la liste de nos matchs tous les bons matchs existant de taille 1 ayant dans leurs ensemble b l'arête de départ.

<u>Déroulement de l'algorithme:</u>

A chaque itération on prend un match et une arête qui n'a pas été encore visitée par l'algorithme pour ce match.
L'association est écrite de la forme a1,a2 (arête de la structure) associé avec b1,b2 (arête du motif) avec a1, a2, b1, b2 des nœuds respectivement de S et M.
On va alors faire la liste de toutes les associations possibles (par encore faite) des arêtes de a1 avec les arêtes de b1. On effectue de même pour a2 et b2.
En faisant un produit de ces deux listes on obtient toutes les combinaisons d'association des arêtes de a1 et a2 avec les arêtes de b1 et b2.
Pour chaque association possible calculé on créer un nouvel agrandissement du match sur lequel on itère en ajoutant aux sous ensembles les nouvelles arêtes associés.

Donc après une itération nous avons donc trouver toutes les possibilités d'agrandissement du match. Si c'est un bon match alors parmis toutes ces possibilités d'agrandissement, il se trouve un bon match de la même solution d'une taille plus grande.

Comme on va réitérer sur ce bon match agrandit de nouveau, alors on va l'agrandir à nouveau. Et ce jusqu'à qu'il soit un bon match avec des ensembles équivalents à ceux de sa solution.

Les mauvais matchs vont eux aussi s'agrandir jusqu'à que de nouveaux agrandissement ne soit possible (soit car on ne peut pas continuer le match, soit parce que les matchs continuer ont un degré de différence trop fort).
Si on peut étirer un mauvais match jusqu'à qu'il parvienne à être un motif approximé de M alors c'est une solution, donc par définition il ne peut être la résultante d'ancien mauvaismatch.

Comme les matchs n'interagissent pas entre eux, et que l'algorithme s'arrête lorsque tous les matchs sont traités, on est assuré de trouver tous les motifs approximatifs de M dans S.

Pseudo-code (graph non orienté, problème des labels uniquement)

<u>Annotation:</u>

Soit:

E_1 l'ensemble d'association traité

E_2 l'ensemble d'association pas encore traité

p la fonciton calculant la penalité d'un ensemble d'association comme la somme de la penalité generé par chacune des associations

<u>Entrée:</u>

S: Graphe de la structure

M: Graphe du motif

n: Pénalité maximum d'arête

<u>Sortie:</u>

L'ensemble des occurrences possible de M dans S au degré de pénalité maximum n près

<u>Execution:</u>

//Initialisation

Soit R un ensemble intialisé comme vide

Soit F une file

Soit $e_0 \in E(M)$ pour chaque arête $e_S \in E(S)$:

enfiler dans D le triplet $(\{\varnothing\},\{(e_0,e_S)\},k)$ avec $k=n-p(\{(e_0,e_S)\})$

//Devellopement

Tant que F n'est pas vide

On défile un triplet (E_1,E_2,k)

Si $E_2 = \varnothing$ et tout les noeuds de E_1 sont présent au moins deux fois dans E_1

$R \cup \{E_1\}$

Sinon

On prend $(e_M,e_S) \in E_2$, on pose $e_M=(u_M,v_M)$ et $e_S=(u_S,v_S)$

Soit A_{Um} l'ensemble des arêtes incidentes à u_M sans e_M

Soit A_{Vm} l'ensemble des arêtes incidentes à v_M sans e_M

Soit A_{Us} l'ensemble des arêtes incidentes à u_S sans e_S

Soit A_{Vs} l'ensemble des arêtes incidentes à v_S sans e_S

Soit P le produit cartésien entre l'ensemble des injections de A_{UM} dans A_{US} et l'ensemble des injections de A_{vm} dans A_{us}

Pour tout élément c appartenant à P

On calcule $k_C=k-p(C)$

Si $k_C>=0$

Enfiler dans F le tripler $(E_1 \cup (e_M,e_S) , (E2\backslash(e_M,e_S) \cup C,k_C)$

//Fin

retourner R

On peut facilement consideré que la penalité est un vecteur de deux entier ce qui nous permet d'avoir une penalité maximal correspondant au nombre d'arrête mal typé (long-range ou short-range) par rapport au motif correspondant. Alors on considere que $k_C>=0$ quand toutes les valeurs de k_c sont supérieure à 0.

p(C) renvoie un vecteur contenant les pénalités d'étiquettage et la pénalité de typage de l'ensemble C.

Pseudo-code (graph orienté, problème des labels uniquement)

Annotation:

Soit:

f la fonction d'étiquetage

E_1 l'ensemble d'association traité

E_2 l'ensemble d'association pas encore traité

p la fonciton calculant la penalité d'un ensemble d'association comme la somme de la penalité generé par chacune des associations

Soit e_M l'arête formé entre les sommets a et b, alors **inv(e_M)** est l'arrete formé entre les sommets b et a

Entrée:

S: Graphe de la structure orienté

M: Graphe du motif orienté

n: Pénalité maximum d'arête

Sortie:

L'ensemble des occurrences possible de M dans S au degré de pénalité maximum n près

Execution:

//Initialisation

Soit R un ensemble intialisé comme vide

Soit F une file

Soit $e_0 \in E(M)$ pour chaque arête $e_S \in E(S)$:

    enfiler dans D le triplet ($\{\varnothing\},\{(e_0,e_S)\},k$) avec $k=n-p(\{f(e_0),f(e_S)\})$

//Devellopement

Tant que F n'est pas vide

    On défile un triplet (E_1,E_2,k)

    Si $E_2 = \varnothing$ et tout les noeuds de E_1 sont présent au moins deux fois dans E_1

        $R \cup \{E_1\}$

    Sinon

        On prend $(e_M,e_S) \in E_2$, on pose $e_M=(u_M,v_M)$ et $e_S=(u_S,v_S)$

        Soit A_{Um} l'ensemble des arêtes entrantes de u_M sans e_M

        Soit A_{Vm} l'ensemble des arêtes entrantes de v_M sans e_M

        Soit A_{Us} l'ensemble des arêtes entrantes de u_S sans e_S

        Soit A_{Vs} l'ensemble des arêtes entrantes de v_S sans e_S

        Soit B_{Um} l'ensemble des arêtes sortante de u_M sans e_M

        Soit B_{Vm} l'ensemble des arêtes sortante de v_M sans e_M

        Soit B_{Us} l'ensemble des arêtes sortante de u_S sans e_S

        Soit B_{Vs} l'ensemble des arêtes sortante de v_S sans e_S

        Soit P le produit carthésien entre

            Le produit carthésien entre l'ensemble des injections de A_{UM} dans A_{US} et l'ensemble des injections de A_{vm} dans A_{us}

            Le produit carthésien entre l'ensemble des injections de B_{UM} dans B_{US} et l'ensemble des injections de B_{vm} dans B_{us}

Pour tout élément c appartenant à P

k_c=k

Pour tout élements a de c:

On pose a=(e_S,e_M)

Si inv(e_M) n'est pas présent dans $E_1 \cup E_2 \cup$ iter(c)

k_C=k_C-p(C)

Sinon si inv(e_M) est présente dans $E_1 \cup E_2 \cup$ iter(c) mais associée avec une autre arête que inv(e_S) ou si inv(e_s) est présente dans dans $E_1 \cup E_2 \cup$ iter(c) mais associée avec une autre arrête que inv(e_M)

On passe au l'element de P suivant

Si k_C>=0

Enfiler dans F le triplet ($E_1 \cup$ (e_M,e_S) , (E2\(e_M,e_S) $\cup$ C,k_C)

//Fin

retourner R

Pseudo-code (graph orienté, problème des labels et arêtes en plus et en moins)

Annotation:
Soit:
f la fonction d'étiquetage
E_1 l'ensemble d'association traité
E_2 l'ensemble d'association pas encore traité
p la fonciton calculant la penalité d'un ensemble d'association comme la somme de la penalité generé par chacune des associations
Soit e_M l'arête formé entre les sommets a et b, alors **inv(e_M)** est l'arrete formé entre les sommets b et a
iter(c) , le sous-ensemble des valeurs de c sur lesquels nous avons dejà iteré
Entrée:
S: Graphe de la structure orienté
M: Graphe du motif orienté
n_1: Pénalité maximum d'arête étiquetté différement
n_2: Pénalité maximum d'arête étiquetté en moins
n_3: Pénalité maximum d'arête étiquetté en plus

Sortie:
L'ensemble des occurrences possible de M dans S au degré de pénalité maximum pres (nombre d'arrête différement étiquetté, nombre d'arrête en plus, nombre d'arrête en moins)
Execution:

//Initialisation
Soit R un ensemble intialisé comme vide
Soit F une file
Soit $e_0 \in E(M)$ pour chaque arête $e_S \in E(S)$:
    enfiler dans D le triplet $(\{\varnothing\},\{(e_0,e_S)\},k)$ avec $k=n-p(\{f(e_0),f(e_S)\})$
//Devellopement
Tant que F n'est pas vide
    On défile un triplet (E_1,E_2,k)
    Si taille($E_1 \cup E_2$) $\in$ [taille(E_M)-n_2;taille(E_M)+n_3] et tout les noeuds de $E_1 \cup E_2$ sont présent au moins deux fois dans E_1
        Si valide($E_1 \cup E_2$, n_2, n_3)
            $R \cup \{E_1 \cup E_2\}$
    Si taille(E_2)!=0 :
        On prend $(e_M,e_S) \in E_2$, on pose $e_M=(u_M,v_M)$ et $e_S=(u_S,v_S)$ tel que $inv(e_M) \notin E_1$
            Soit A_{Um} l'ensemble des arêtes entrantes de u_M sans e_M
            Soit A_{Vm} l'ensemble des arêtes entrantes de v_M sans e_M
            Soit A_{Us} l'ensemble des arêtes entrantes de u_S sans e_S
            Soit A_{Vs} l'ensemble des arêtes entrantes de v_S sans e_S
            Soit B_{Um} l'ensemble des arêtes sortante de u_M sans e_M
            Soit B_{Vm} l'ensemble des arêtes sortante de v_M sans e_M
            Soit B_{Us} l'ensemble des arêtes sortante de u_S sans e_S

Soit B_{Vs} l'ensemble des arêtes sortante de v_S sans e_S

Soit P le produit carthésien entre:

-Le produit carthésien entre l'ensemble des injections de A_{UM} dans A_{US} et l'ensemble des injections de A_{vm} dans A_{us}

-Le produit carthésien entre l'ensemble des injections de B_{UM} dans B_{US} et l'ensemble des injections de B_{vm} dans B_{us}

Pour tout élément c appartenant à P

k_c=k

Pour tout élements a de c:

On pose a=(e_S,e_M)

Si inv(e_M) n'est pas présent dans $E_1 \cup E_2 \cup$ iter(c)

k_C=k_C-p(C)

Sinon si inv(e_M) est présente dans $E_1 \cup E_2 \cup$ iter(c) mais associée avec une autre arête que inv(e_S) ou si inv(e_s) est présente dans dans $E_1 \cup E_2 \cup$ iter(c) mais associée avec une autre arrête que inv(e_M)

On passe au l'element de P suivant

Si k_C>=0

Enfiler dans F le tripler ($E_1 \cup$ (e_M,e_S) , (E2\(e_M,e_S) $\cup$ C,k_C)

//Fin

retourner R

Fonction de validation:

<u>Entrée:</u>

E1: Un ensemble d'association d'arrête du graph de la structure

S: graph de la structure

M: graph du motif

n_2: penalité max du nombre d'arrête en plus

n_3: penalité max du nombre d'arrête en moins

<u>Sortie:</u>

True si E_1 est un match respectant les penalités n_2 et n_3

False si E_1 est un match respectant les penalités n_2 et n_3

<u>Execution:</u>

On defini S l'ensemble des sommtes v_S tels que $v_s \in E_1 \cap V(S)$ avec l'ensemble des sommets v_M tel que $v_M \in E_1 \cap V(M)$

Si un sommet v_S est associé à plusieurs sommet V_M ou inversement

retourner Faux

Sinon

pour chaque couple de somment (v_s,v_M) $\in$ S:

pour chaque couple de sommet (v'_S,v'_M) $\in$ S\{(v_s,v_M):

si (v_S,v'_S) $\in$ E(S):

Si (v_M,v'_M) $\notin$ E(M):

n2=n2-1

Si n2<0

```
                  retourner Faux
      si (v_S,v'_S) ∉ E(S):
            Si (v_M,v'_M) ∈ E(M):
                  n3=n3-1
                  Si n3<0
                        retourner Faux
retourner True
```