Décomposition de matroïdes orientés:

Feuille de route programme 1 trouver-sous-modèles fixes.

MARIN Yann

4 février 2022

Ceci est un compte-rendu des capacités d'un programme pour résoudre le problème de trouver les sous-modèles fixes maximaux. Il fait suite à deux compte-rendus et une autre feuille de route disponnibles ici : https ://github.com/YannMarin/stage-info (branch master) .
Rappellons le problème donné :
Etant donné E et X, trouvé tous les P tels que $S_P \subseteq X$.
Il y à plusieurs façons de résoudre ce problème selon ce que l'on sait sur E,X,$\overline{X}$ et nos présomptions sur L et E-L. Dans tous les cas, nous sommes obligés de tester une propriété sur un très grand nombre de sous ensembles de E ou de X. Il nous faut donc trouver la manière la plus adapté.

Table des matières

0.1 Glossaire . . . 1

1 L'algorithme **2**
1.1 Convergence : . . . 2
1.2 Correction : . . . 3
1.3 Ordre de parcourt . . . 3
1.4 Temps de calcul . . . 3
1.5 Espace de calcul . . . 3
1.5.1 Améliorations . . . 3
1.6 Explosion en temps : . . . 3
1.6.1 Améliorations . . . 4
1.7 Resultat du programme . . . 4
1.7.1 Améliorations . . . 4
1.8 Algorithme a comparer . . . 4

0.1 Glossaire

— $E = \{1 :: n\}$ ensemble de n points.
— d la dimension d'un espace affine, par défaut d=4.
— B l'ensemble des $\binom{n}{d}$ d-uplets sur E.
— $X \subseteq B$ un ensemble de d-uplets.
— $P \subseteq E$ un ensemble de points.
— $S_P \subseteq B$ l'ensemble des $\binom{|P|}{d}$ d-uplets dans P.
— $L = \{P \subseteq E | S_P \subseteq X\}$

— $\overline{X} = E - X$
— $\overline{L} = \{P \subseteq E | \exists x \in \overline{X} \text{ tq } x \in S_P\}$
— $max(L) = \{P \in L| \text{ si } P \subseteq P' \in L \text{ alors } P = P'\}$
— $X_P = X \cap S_P$
— PS : Chaque itération du mot "Fixe" est à remplacer par le mot "constant".

1 L'algorithme

Pour résoudre ce problème, de nombreux algorithmes sont possibles dont une liste non exhaustive se trouve sur le même git. (EN COURS D'ECRITURE), chacun de ces algorithmes sont plus ou moins efficaces selon la situation.

L'algorithme que l'on regarde est le suivant :

Algorithm 1 TROUVER-SOUS-MODELES-CONSTANTS

```
Require: X, E
Ensure: L
  FIFO=[(x,X-x) pour tout x ∈ X]
  while |FIFO| > 0 do
      P = FIFO[0][0]
      X = FIFO[0][1]
      p=P[-1] (le dernier élément de P).
      bool-est-max = true
      for i de p+1 à n do
          compteur = 0
          for x ∈ X do
              if x ∈ S_P then
                  compteur +=1
                  X=X-x
              end if
          end for
          if C(|P|, d) + compteur = C(|P|+1, d) then
              bool-est-max = false
              FIFO = FIFO ∪ (P ∪ p, X)
          end if
      end for
      if bool-est-max = true then
          L = L ∪ P
      end if
      FIFO = FIFO-FIFO[0]
  end while
```

Où FIFO est une liste First-in First-Out.

1.1 Convergence :

FIFO n'augmente que lorsque l'on trouve des sous-modèles constants, comme les sous-modèles constants augmentent ou sont retiré de FIFO, l'algorithme fini par converger.

1.2 Correction :

Si un sous-modèle est non constant, tout sous-modèles qui le contient est non fixe. Si un sous modèle est non constant, l'algorithme ne test pas les sous-ensembles qui le contiennent.
Si un sous-modèle est constant, l'algorithme visite tous les sous-ensembles qui le contient et qui l'ont comme préfixe.
Si un sous-modèles est constant, alors tous sous-ensemble de ce sous-modèle est constant. Donc son préfixe de taille d est un sous-modèle constant de taille minimal.
L'algorithme trouve donc bien tous les sous-modèles fixes maximaux (En plusieurs fois parfois).
Il trouve cependant aussi des sous-modèles fixes non maximaux. Un trie est nécessaire à la fin.

1.3 Ordre de parcourt

On parcourt exactement tous les sous-ensemble de E qui sont après X[0] dans l'ordre léxicographique, où X[0] est le premier d-uplet constant dans l'ordre lexicographique, et qui sont contenus dans un sous-modèles constants maximum, une et une seule fois.

1.4 Temps de calcul

On calcul chaque couple $(P \subseteq E, X_P \subseteq X)$ à partir d'un couple $(P - p, X_{P-p}))$ en calculant $(X - X_P) \cap S_{P-p}$.
Le test pour savoir si un sous-modèles est constant à partir de $(X - X_P) \cap S_{P-p}$ est en O(1) grâce à un tableau des $\binom{n}{k}$.
En notant E[<= t] l'ensemble des sous-ensemble de E de taille <= t alors on peut majorer le nombre d'itérations par E[<= $t_max + 1$]$|X|$

1.5 Espace de calcul

On doit enregistrer L.
Pour passer de l'indice d'un d-uplet à ses points on a une liste de taille $\binom{n}{k}$ de d-uplets.
Au maximum, $|FIFO| = \binom{n}{k}$ pour un certain k.
Chaque $f = (P, X_P) \in FIFO$ contient un ensemble de sommet ¡n et un ensemble de d-uplet $< ||X|$.
A tout instant le nombre d'entier enregistré est donc majoré par $O(\binom{k}{n}(n + |X|)$

1.5.1 Améliorations

On pourrait diminuer le nombre de bit pour écrire un entier, mais les indices des bases vont jusqu'à $\binom{n}{d}$.
On pourrait ne rien enregister mais alors le temps de calcul serait en E[<= $t_max + 1$]$|X|$.

1.6 Explosion en temps :

Si le modèle à beaucoup de points.
Si le modèle à beaucoup de sous-modèles constants.
Si la taille maximal des sous-modèles constants maximum est grande.

1.6.1 Améliorations

Lancer l'algorithme sur les $\binom{n}{k}$ sous-ensembles de taille k au lieu des d-uplets de X pour k proche de la taille des sous-modèles constants maximum maximaux.

1.7 Resultat du programme

Sur 16,20 et 23 points il est presque instantané, sur 25 points il prend quelques minutes, sur 30 points il prend quelques heures.
Le programme dépend énormément de la taille maximal des sous-modèles constants maximum.

1.7.1 Améliorations

Changer le nombre de bit par int.
Passer en C++.
Avoir plus d'information au préalable sur le modèle.

1.8 Algorithme a comparer

Il faudrait comparer l'efficacité de ce programme avec ces deux mêmes versions mais qui n'enregistrent rien en espace.