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-rendu 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

— $E = \{1 :: n\}$ 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.
— $L = \{P \subseteq E | S_P \subseteq X\}$
— $\overline{X} = E - X$
— $\overline{L} = \{P \subseteq E | \exists x \in \overline{X} \text{ tq } x \in S_P\}$
— $max(L) = \{P \in L | \text{ si } P \subseteq P' \in L \text{ alors } P = P'\}$
— $X_P = X \cap S_P$

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-FIXES

Require: X, E
Ensure: L
  FIFO=[(E,X)]
  while $|FIFO| > 0$ **do**
    for tout les sous ensembles P_i de FIFO[0][0] qui ne sont pas déjà dans FIFO **do**
      On calcul $X_i = X \cap S_{p_i}$
      if $|X_i| = \binom{|P_i|}{d}$ **then**
        $L = L \cup P_i$
      else
        $FIFO = FIFO \cup (P_i, X_i)$
      end if
    end for
    $FIFO = FIFO - FIFO[0]$
  end while

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

1.1 Convergence :

Une fois que l'algorithme à tester les 2^n (au maximum), FIFO est vide et l'algorithme s'arrête.

1.2 Correction :

L'algorithme visite tous les sous-modèles fixes maximaux. De plus, si P est un sous-modèles fixe maximal alors $|X \cap S_P| = \binom{|P|}{d}$ et P est ajouté a L.
Si P n'est pas un sous-modèle fixe, alors $|X \cap S_P|! = \binom{|P|}{d}$ et P n'est pas ajouté à L.

1.3 Ordre de parcours :

On parcourt de manière unique (une condition supplémentaire non présente dans l'algorithme l'assure dans le programme) tous les $P \subseteq E$ dans l'ordre des $\binom{n}{k}$ avec k décroissant et

dans l'ordre léxicographique.

De manière exacte, on test tous les $P \subseteq E$ qui ne sont pas contenus dans un sous-modèle fixe antérieur dans l'ordre léxicographique.

1.3.1 Amélioration

Si on était capable de résoudre le problème 1 du CR2, alors on pourrait ne tester que les sous-ensembles qui ne sont pas des sous-modèles fixes.

1.4 Temps de calcul

Pour chaque sous-ensemble $P \subseteq E$ on, calcul les $P_i \subseteq P$ tel que $P_i < P$ dans l'ordre lexicographique et pour chaque P_i on fais deux étapes :

1.4.1 Etape 1 : On calcul $X_{P_i} = X_P \cap S_{P_i}$

C'est l'étape la plus longue. Pour chaque $x \in X_P$ on doit tester si $p = P - P_i \notin x$.
On peut majorer ce temps de calcul par O($|X|$).

1.4.2 Etape 2 : On test si P_i est un sous-modèles fixe

On doit tester si $X_{P_i} = \binom{|P_i|}{d}$ En utilisant un tableau pour enregister les différentes tailles de S_{P_i} en fonction des tailles des P_i cette étape est en O(1).

On peut majorer le temps de calcul global par $\overline{L}|X|$.

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 $\overline{L}|X|$

1.6 Explosion en temps :

Si le modèle à beaucoup de points.
Si le modèle à peux de sous-modèles constants.
Si $X \cap S_p \simeq S_P$ pour chaque $P \subseteq E$

1.6.1 Améliorations

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

1.7 Explosion en espace

S'il existe une partie des derniers sous-ensemble de taille k et des premiers sous-ensemble de taille k+1 qui n'ont pas de sous-modèles fixes et qui dépasse la mémoire alloué.

1.7.1 Amélioration

Commencer l'algorithme avec les sous-ensembles de taille $j > k + 1$.

1.8 Resultat du programme

Sur 16 points il est presque instantané, sur 20 points il prend moins de 2 minutes, sur 23 points il prend une heure, sur 25 points la mémoire explose avant de trouver le premier sous-modèle fixe.

1.8.1 Améliorations

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

1.9 Algorithme a comparer

Il faudrait comparer l'efficacité de ce programme avec les deux algorithmes suivants :
-Un qui n'enregistre par les P et X_P.
-Un qui commence par des $\binom{n}{4}$ d-uplets et cherche les sous-modèles constants maximaux en ajoutant des sommets et fait le test à partir de $\overline{X}$ -taille ¡k, taille ¿k