Les méthodes récentes d'économie de mémoire s'appliquent-elles à un problème P-complet ?

Projet Honor - IFT4055

Ahmed Mhedhbi

Superviseur: Prof. Pierre McKenzie

May 3, 2025

Université de Montréal

Sommaire

Pourquoi étudier l'espace catalytique?

- Économie de mémoire: comprendre les limites fondamentales de l'espace est aussi crucial que l'étude du temps d'exécution.
- *L vs P*: comme la question *P vs NP*, le problème *L vs P* (log-espace versus temps polynomial) demeure ouvert.
- le calcul catalytique offre une nouvelle perspective : un nouveau paradigme pour gagner de l'espace.
- Objectif du projet : tester si ces techniques d'économie de mémoire s'étendent aux problèmes P-complets.

Sommaire

STCONN et espace catalytique

STCONN: décider si $s \rightsquigarrow t$ dans un digraphe à $|V| = n$.

- *Savitch (1970)*: $O(\log^2 n)$ espace (démonstration générique).
- *Reingold (2008)*: $O(\log n)$ espace, cas non orienté ($\mathsf{L} = \mathsf{SL}$).
- *NNJAG* (Poon-1993, Edmonds-1999) : best $\approx n/2^{\Theta(\sqrt{\log n})}$ espace (polytemps).
- *Catalytique (Buhrman et al. 2014)*: STCONN en temps $O(n^9)$, $O(\log n)$ espace interne $+\ O(n^2 \log n)$ catalytique.

TreeEval

TreeEval: évaluer un arbre complet binaire de hauteur h, alphabet de taille k.

- Structure de l'arbre :
  - Noeuds internes : des fonctions $f : [k] \times [k] \to [k]$.
  - Feuilles : des valeurs dans $[k]$.
- Algorithme de base : parcours en profondeur, nécessite $O(h \log k)$ bits en espace, impliquant une profondeur $\Omega(h \log k)$.
- *Cook–Mertz (2020)*: $O(h \log k / \log h)$
- *Cook–Mertz (2024)*: $O(h \log \log k + \log k)$ espace

Simulation temps $\rightarrow$ espace

Williams (2025): toute une machine de Turing temps $t(n)$ se simule en espace $\sqrt{t(n)\log t(n)}$.

$$\mathrm{TIME}[t] \;\subseteq\; \mathrm{SPACE}[\sqrt{t \log t}]$$

$$\text{implications : } \begin{cases} \text{circuits de taille } s \text{ en espace } \sqrt{s}\, \textit{poly} \log(s) \\ \text{Circuits de taille } s \;\; \rightsquigarrow \;\; \text{BP de taille } 2^{O(\sqrt{s}\ \log^k s)}, \end{cases}$$

Établit un lien avec l'analyse des programmes de branchement.

Note : ce théorème est paru après le démarrage de ce projet, mais il mérite une place centrale : seul progrès majeur en économie d'espace depuis 1977 (Best Paper STOC 2025), et il peut être reconnecté à l'étude de GEN, notre problème P-complet cible.

Sommaire

Intuition de l'algorithme TreeEval

- Reprend l'approche de Cook–Mertz (2024) : transformer l'évaluation d'un arbre complet en somme de polynômes sur un corps fini à l'aide des racines de l'unité.
- Formule clé :

$$\sum_{j=1}^{m} -1 \cdot p\big(\omega^j \tau_1 + x_1,\ \ldots,\ \omega^j \tau_n + x_n\big) \;=\; p(x_1, \ldots, x_n)$$

- Calcul "gout à gout" (bit à bit) pour contourner les restrictions.
- Construction d'un *register program* uniforme :
  - chaque nœud appelle récursivement deux sous-programmes (ses enfants),
  - applique une somme catalytique de m itérations pour calculer la valeur courante,
  - puis nettoie l'état auxiliaire derrière le calcul pour revenir à l'état initial.

Implémentation de TreeEval (Cook–Mertz 2024)

- **Prototype Python "warm-up "**
  - Utilisation de la bibliothèque `Galois`.
  - Simulation des *register programs*.
- **Difficultés**
  - Préparation et configuration de l'environnement pour faire le calul.
  - Vérification de la correction de l'algorithme : chaque appel engendre 4 appels récursifs (2 à la même fonction, 2 à une fonction auxiliaire).
  - assurer la complexité de chaque opération.
- **Passage à C++**
  - Représentation des éléments de $\mathbb{GF}(2^m)$ par masques de bits et opérations bit à bit.
  - Optimisation des boucles et de l'arithmétique des corps finis.
- **Constats et apprentissages**
  - Auto-formation intensive en théorie des corps finis.
  - L'implémentation concrète a renforcé ma compréhension de l'algorithme.

Sommaire

Définition de GEN

Définition

Entrée : $g\colon [n] \times [n] \to [n]$ (triples $i * j = g(i,j)$).

Sortie : `accepter` si $n \in \langle\{1\}\rangle_g$ (fermeture de 1 sous $*$).

GEN est dans P

Entrée: $g : [n] \times [n] \to [n]$

Sortie: accept si $n \in \langle\{1\}\rangle_g$, sinon reject

$T := \{1\}$

tant que $\exists\, i, j \in T$ tel que $k := g(i, j) \notin T$ **faire**

$T := T \cup \{k\}$

fin tant que

accepter si $n \in T$

Complexité:

- *Temps.* $\mathcal{O}(n^3)$: chaque itéation on parcourt au pire n^2 éléments et on ajoute au pire n éléments
- *Espace.* $\mathcal{O}(n)$: un tableau de marquage de taille n, chaque élément contient un seul bit.

GEN est P-complet

P-complétude (Jones–Laaser 1977)

GEN est P-complet sous la réduction en L (logspace).

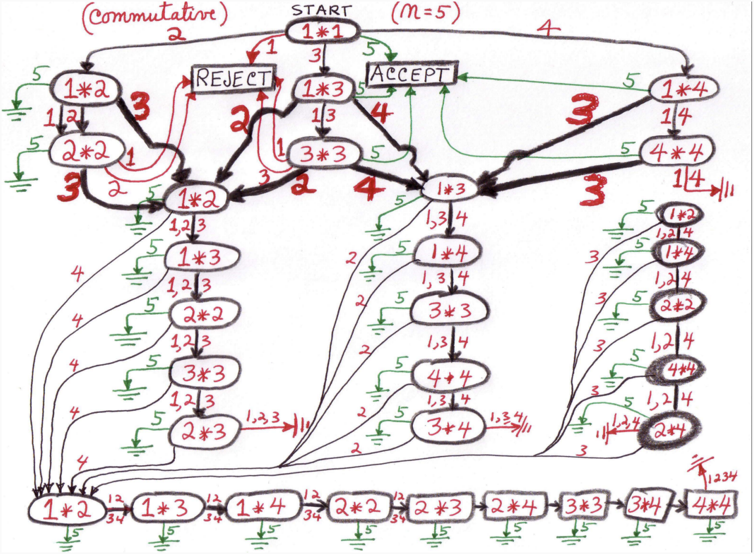

Figure 1: Un exemple de programme de branchement pour GEN (commutatif) pour $n = 5$.

Programmes de branchement pour GEN

- *Gál–Koučký–McKenzie (2008)*: Lower-bounds exponentielles pour BPs incrémentaux

Approches utilisés

- Compréhension approfondie des résultats pour GEN (Gál–Koučký–McKenzie 2008) — nécessité de décortiquer la structure pour toute amélioration.
- Subdivision le circuit pour GEN en *couches*: — évaluer l'impact.
- Combinaison de l'algorithme Cook–Mertz (2024) pour TreeEval: — exploration de nouveaux équilibres temps/espace.
- Intégration de la simulation temps$\rightarrow$ espace de Williams (2025) pour accélérer GEN.
- Analyse préliminaire de l'effet sur la séparation P vs PSPACE — réflexion sur les limites et les gains possibles.

Sommaire

Conclusion

- Peut-on combiner l'algo de Cook–Mertz & pebbling ?
- Intégrer la simulation $\sqrt{t \log t}$ de Williams en accélérant un algorithme pour GEN ?
- Impact potentiel sur la séparation P vs PSPACE.
- La question reste toujours ouverte.

Merci pour votre attention.

Questions?