Séminaire BOURBAKI Janvier 2017
69e année, 2016–2017, no 1125, p. 135 à 182
doi:10.24033/ast.1063

ISOMORPHISMES DE GRAPHES EN TEMPS QUASI-POLYNOMIAL
[d'après Babai et Luks, Weisfeiler–Leman, ...]

par **Harald Andrés HELFGOTT**

1. INTRODUCTION

Soient $\mathbf{x}$, $\mathbf{y}$ deux chaînes de caractères, à savoir, deux applications $\Omega \to \Sigma$, où Σ (l'*alphabet*) et Ω (le *domaine*) sont des ensembles finis. Tout groupe de permutations [(1)] $G < \mathrm{Sym}(\Omega)$ agit sur l'ensemble Σ^Ω des chaînes de domaine Ω sur un alphabet Σ. Pour nous, *décrire un groupe* G, ou *être donné un groupe* G, voudra toujours dire « donner, voire être donné, un ensemble de générateurs de G » ; *décrire une classe* $H\pi$ voudra dire « donner un élément π de la classe et un ensemble de générateurs de H ».

Le *problème de l'isomorphisme de chaînes* consiste à déterminer, étant donnés $\mathbf{x}$, $\mathbf{y}$ et G, s'il y a au moins un élément π de G qui envoie $\mathbf{x}$ sur $\mathbf{y}$, et, si de tels éléments (*isomorphismes*) existent, à les décrire. Il est clair que l'ensemble des isomorphismes $\mathrm{Iso}_G(\mathbf{x}, \mathbf{y})$ forme une classe $\mathrm{Aut}_G(\mathbf{x})\pi$ du groupe $\mathrm{Aut}_G(\mathbf{x})$ d'automorphismes de $\mathbf{x}$ dans G, c'est-à-dire du groupe consistant dans les éléments de G qui envoient $\mathbf{x}$ sur lui-même.

Le défi consiste à donner un algorithme qui résolve le problème en temps polynomial en la taille $n = |\Omega|$ de Ω, voire en temps raisonnable. Par exemple, le temps employé pourrait être *quasi-polynomial* en n, ce qui veut dire $\exp\left(O(\log n)^{O(1)}\right)$. Ici, comme toujours, $O(f(n))$ désigne une quantité bornée par $C \cdot f(n)$, pour n assez grand et $C > 0$ une constante, et O_ϵ indique que la constante C dépend de ϵ.

Une grande partie de la motivation pour le problème de l'isomorphisme de chaînes vient du fait que le *problème de l'isomorphisme de graphes* se réduit à lui. Ce problème consiste à déterminer si deux graphes finis Γ_1 et Γ_2 sont isomorphes, et, s'ils le sont, à décrire la classe de leurs isomorphismes. (Un *isomorphisme* $\pi : \Gamma_1 \to \Gamma_2$ est une bijection π de l'ensemble de sommets de Γ_1 vers celui de Γ_2 telle que $\pi(\Gamma_1) = \Gamma_2$.)

(1) Pour nous, $G < S$ (ou $S > G$) veut dire « G est un sous-groupe de S, pas forcement propre. »

Une solution permettrait, par exemple, de trouver une molécule dans une base de données.

Le problème de l'isomorphisme de graphes se réduit en temps polynomial au problème de l'isomorphisme de chaînes, de la façon suivante. Supposons sans perte de généralité que Γ_1 et Γ_2 ont le même ensemble de sommets V. Alors, nous pouvons définir Ω comme l'ensemble des paires d'éléments de V (ordonnés ou non ordonnés, suivant que nos graphes sont orientés ou pas). La chaîne $\mathbf{x}_i$, $i = 1, 2$, est définie comme suit : pour la paire $a = \{v_1, v_2\}$ (ou $a = (v_1, v_2)$, si nos graphes sont orientés), la valeur de $\mathbf{x}_i(a)$ est 1 s'il y a une arête entre v_1 et v_2 en Γ_1, et 0 dans le cas contraire. Soit G l'image de l'homomorphisme $\iota : \mathrm{Sym}(V) \to \mathrm{Sym}(\Omega)$ définie par $\sigma^\iota(\{v_1, v_2\}) = \{\sigma(v_1), \sigma(v_2)\}$, où $\sigma^\iota = \iota(\sigma)$. Alors ι induit une bijection entre la classe des isomorphismes de Γ_1 à Γ_2 et la classe $\mathrm{Iso}_G(\mathbf{x}_1, \mathbf{x}_2)$.

THÉORÈME 1.1 (Babai). — *Le problème de l'isomorphisme de chaînes $\Omega \to \Sigma$ peut être résolu en temps quasi-polynomial en le nombre d'éléments du domaine Ω.*

En novembre 2015, Babai a annoncé une solution en temps quasipolynomial, avec un algorithme explicite. La préparation de cet exposé m'a conduit à trouver une erreur non triviale dans l'analyse du temps, mais Babai a réussi à le réparer en simplifiant l'algorithme. La preuve est maintenant correcte.

COROLLAIRE 1.2 (Babai). — *Le problème de l'isomorphisme de graphes peut être résolu en temps quasi-polynomial en le nombre de sommets.*

Notre référence principale sera **[3]** ; nous nous servirons aussi de la version courte **[2]**. Nous essayerons d'examiner la preuve de la façon la plus détaillée possible dans un exposé de ce format, en partie pour aider à éliminer tout doute qui pourrait rester sur la forme actuelle du résultat.

La meilleure borne générale connue antérieurement pour le temps requis par le problème de l'isomorphisme de graphes, due à Luks **[5]**, était $\exp(O(\sqrt{n \log n}))$,

L'usage de la *canonicité* joue un rôle crucial dans la stratégie de Babai. Comme dans la théorie de catégories, voire dans l'usage courant, un choix est *canonique* s'il est fonctoriel. La situation typique pour nous sera la suivante : un groupe $G < \mathrm{Sym}(\Omega)$ agit sur Ω, et donc sur Σ^Ω ; il agit aussi sur un autre ensemble S, et donc aussi sur les applications $S \to \mathscr{C}$, où $\mathscr{C}$ est un ensemble fini. Une application $S \to \mathscr{C}$ s'appelle un *coloriage* ; l'ensemble $\mathscr{C}$ s'appelle l'ensemble de *couleurs*. Un choix *canonique* (en relation à G) d'un coloriage de Ω pour chaque chaîne $\mathbf{x} \in \Sigma^\Omega$ est une application qui va de Σ^Ω aux coloriages et qui commute avec l'action de G.

En particulier, un choix canonique peut être un outil pour détecter des non-isomorphismes : si les coloriages $C(\mathbf{x})$ et $C(\mathbf{y})$ induits canoniquement par $\mathbf{x}$ et $\mathbf{y}$ ne

sont pas isomorphes l'un à l'autre – par exemple, s'ils ont un nombre différent d'éléments vermeils – alors $\mathbf{x}$ et $\mathbf{y}$ ne sont pas isomorphes l'un à l'autre. Même quand il y a des isomorphismes dans G qui envoient $C(\mathbf{x})$ sur $C(\mathbf{y})$, la classe $\mathrm{Iso}_G(C(\mathbf{x}), C(\mathbf{y}))$ de tels isomorphismes sert à délimiter la classe d'isomorphismes $\mathrm{Iso}_G(\mathbf{x}, \mathbf{y})$ de $\mathbf{x}$ à $\mathbf{y}$, puisque cette dernière est forcément un sous-ensemble de $\mathrm{Iso}_G(C(\mathbf{x}), C(\mathbf{y}))$.

La preuve assimile aussi plusieurs idées développées lors d'approches antérieures au problème. La première étape de la procédure consiste à essayer de suivre ce qui est en essence l'algorithme de Luks [**15**]. Si cet algorithme s'arrête, c'est parce qu'il s'est heurté contre un quotient H_1/H_2 isomorphe à $\mathrm{Alt}(\Gamma)$, où $H_2 \triangleleft H_1 < G$ et Γ est plutôt grand.

Notre tâche majeure consiste à étudier ce qui se passe à ce moment-là. La stratégie principale sera de chercher à colorier Γ d'une façon qui dépend canoniquement de $\mathbf{x}$. Cela limitera les automorphismes et isomorphismes possibles à considérer. Par exemple, si la moitié de Γ est coloriée en rouge et l'autre en noir, le groupe d'automorphismes possibles se réduit à $\mathrm{Sym}(|\Gamma|/2) \times \mathrm{Sym}(|\Gamma|/2)$. Un coloriage similaire induit par $\mathbf{y}$ limite les isomorphismes aux applications qui alignent les deux coloriages. Nous trouverons toujours des coloriages qui nous aident, sauf quand certaines structures ont une très grande symétrie, laquelle, en revanche, permettra une descente à Ω considérablement plus petit. Cette double récursion – réduction du groupe H_1/H_2 ou descente à des chaînes considérablement plus courtes – résoudra le problème.

2. FONDEMENTS ET TRAVAUX PRÉCÉDENTS

En suivant l'usage courant pour les groupes de permutations, nous écrirons r^g pour l'élément $g(r)$ auquel $g \in \mathrm{Sym}(\Omega)$ envoie $r \in \Omega$. Étant donnés une chaîne $\mathbf{x} : \Omega \to \Sigma$ et un élément $g \in \mathrm{Sym}(\Omega)$, nous définissons $\mathbf{x}^g : \Omega \to \Sigma$ par $\mathbf{x}^g(r) = \mathbf{x}\left(r^{g^{-1}}\right)$.

Par contre, nous écrivons Ω^k pour l'ensemble des $\vec{x} = (x_1, \ldots, x_k)$ avec l'action à gauche donnée par $(\phi(\vec{x}))_r = \vec{x}_{\phi(r)}$. L'idée est que ceci est défini non pas seulement pour ϕ une permutation, mais pour toute application $\phi : \{1, \ldots, k\} \to \{1, \ldots, k\}$, même non injective. Nous appelons les éléments de Ω^k *tuples* plutôt que *chaînes*.

2.1. Algorithmes de base

2.1.1. *Schreier-Sims.* — Plusieurs algorithmes essentiels se basent sur une idée de Schreier [**20**]. Il a remarqué que, pour tout sous-groupe H d'un groupe G et tout sous-ensemble $A \subset G$ qui engendre G et contient des représentants de toutes les classes de H dans G,

$$A' = AAA^{-1} \cap H = \left\{\sigma_1\sigma_2\sigma_3^{-1} : \sigma_i \in A\right\} \cap H$$

est un ensemble de générateurs de H.

L'étape suivante est celle de Sims [**21**], [**22**], qui a montré l'utilité de travailler avec un groupe de permutations $G < \mathrm{Sym}(\Omega)$, $\Omega = \{x_1, \dots, x_n\}$, en termes d'une *chaîne de stabilisateurs*

$$G = G_0 > G_1 > G_2 > \dots > G_{n-1} = \{e\},$$

où $G_k = G_{(x_1,x_2,\dots,x_k)} = \{g \in G : \forall 1 \le i \le k \;\; x_i^g = x_i\}$ (*stabilisateur de points*).

L'algorithme de Schreier-Sims (Algorithme 1 ; description basée sur [**15**, §1.2]) construit des ensembles C_i de représentants de G_i/G_{i+1} tels que $\bigcup_{i \le j < n-1} C_j$ engendre G_i pour tout $0 \le i < n-1$. Le temps pris par l'algorithme est $O(n^5 + n^3|A|)$, où A est l'ensemble de générateurs de G qui nous est donné : la fonction FILTRE prend $O(n)$ de temps, et tout g pour lequel elle est appelée satisfait $g \in AC \cup CA \cup C^2$, où C est la valeur de $\bigcup_i C_i$ à la fin de la procédure. Bien sûr, $|C| \le n(n+1)/2$.

Grâce à l'algorithme lui-même, nous pourrons toujours supposer que nos ensembles de générateurs sont de taille $O(n^2)$. Le temps pris par l'algorithme est donc $O(n^5)$. [(2)]

Une fois les ensembles C_i construits, il devient possible d'accomplir plusieurs tâches essentielles rapidement.

EXERCICE 2.1. — *Montrer comment accomplir les tâches suivantes en temps polynomial, étant donné un groupe $G < \mathrm{Sym}(\Omega)$, $|\Omega| = n$:*

(a) *Déterminer si un élément $g \in \mathrm{Sym}(\Omega)$ est dans G.*
(b) *Étant donnés un homomorphisme $\phi : G \to \mathrm{Sym}(\Omega')$, $|\Omega'| \ll |\Omega|^{O(1)}$, et un sous-groupe $H < \mathrm{Sym}(\Omega')$, décrire $\phi^{-1}(H)$.*
(c) [**12**] *Soit $H < G$ avec $[G : H] \ll n^{O(1)}$. Étant donné un test qui détermine en temps polynomial si un élément $g \in G$ appartient à H, décrire H.* Astuce : *travailler avec $G > H > H_1 > H_2 > \cdots$ à la place de $G = G_0 > G_1 > G_2 > \cdots$.*

Ici, comme toujours, « décrire » veut dire « trouver un ensemble de générateurs », et un groupe nous est « donné » si un tel ensemble nous est donné.

L'algorithme de Schreier-Sims décrit le stabilisateur de points $G_{(x_1,\dots,x_k)}$ pour $x_1, \dots, x_k \in \Omega$ arbitraires. Par contre, nous ne pouvons pas demander allègrement un ensemble de générateurs d'un *stabilisateur d'ensemble* $G_{\{x_1,\dots,x_k\}} = \{g \in G : \{x_1^g, \dots, x_k^g\} = \{x_1, \dots, x_k\}\}$ pour G, x_i arbitraires : faire ceci serait équivalent à résoudre le problème de l'isomorphisme lui-même.

(2) Nous supposons que l'ensemble de générateurs initial, spécifiant le groupe G du problème, est de taille $O(n^C)$, C une constante. Le temps pris par la première utilisation de l'algorithme est donc $O\left(n^{\max(5,3+C)}\right)$.

Algorithme 1 Schreier-Sims : construction d'ensembles C_i

fonction SCHREIERSIMS(A, $\vec{x}$) ▷ A engendre $G < \mathrm{Sym}(\{x_1, \dots, x_n\})$
assure $\bigcup_{i \le j < n-1} C_j$ engendre G_i et $C_i \mapsto G_i/G_{i+1}$ est injectif $\forall i \in \{0, 1, \dots, n-2\}$
  $C_i \leftarrow \{e\}$ pour tout $i \in \{0, 1, \dots, n-2\}$
  $B \leftarrow A$
  tantque $B \neq \emptyset$
    Choisir $g \in B$ arbitraire, et l'enlever de B
    $(i, \gamma) \leftarrow$ FILTRER(g, (C_i), $\vec{x}$)
    si $\gamma \neq e$ **alors**
      ajouter γ à C_i
      $B \leftarrow B \cup \bigcup_{j \le i} C_j \gamma \cup \bigcup_{j \ge i} \gamma C_j$
  retourner (C_i)
fonction FILTRER(g, (C_i), $\vec{x}$) ▷ retourne (i, γ) tel que $\gamma \in G_i$, $g \in C_0 C_1 \cdots C_{i-1} \gamma$
requiert $C_i \subset G_i$ et $C_i \to G_i/G_{i+1}$ injectif $\forall i \in \{0, 1, \dots, n-2\}$
assure $g \notin C_0 C_1 \cdots C_i G_{i+1}$ sauf si $(i, \gamma) = (n-1, e)$
  $\gamma \leftarrow g$
  pour $i = 0$ **jusqu'à** $n-2$
    si $\exists h \in C_i$ tel quel $x_{i+1}^h = x_{i+1}^\gamma$ **alors**
      $\gamma \leftarrow h^{-1}\gamma$
    sinon
      retourner (i, γ)
  retourner $(n-1, e)$

2.1.2. *Orbites et blocs.* — Soit donné, comme toujours, un groupe de permutations G agissant sur un ensemble fini Ω. Le domaine Ω est l'union disjointe des *orbites* $\{x^g : g \in G\}$ de G. Ces orbites peuvent être déterminées en temps polynomial [(3)] en $|\Omega|$. Ceci est un exercice simple. La tâche se réduit à celle – simple elle aussi – de trouver les composantes connexes d'un graphe.

Supposons que l'action de G soit transitive. (Il y a donc une seule orbite.) Un *bloc* de G est un sous-ensemble $B \subset \Omega$, $B \notin \{\emptyset, \Omega\}$, tel que, pour $g, h \in G$ quelconques, soit $B^g = B^h$, soit $B^g \cap B^h = \emptyset$. La collection $\{B^g : g \in G\}$ (*système de blocs*) pour B donné partitionne Ω. L'action de G est *primitive* s'il n'y a pas de blocs de taille > 1 ;

[(3)] Pour être précis : $O\left(|\Omega|^{O(1)} + |A||\Omega|\right)$, où A est la taille de l'ensemble de générateurs de G qui nous est donné. Nous omettrons toute mention de cette taille par la suite, puisque, comme nous l'avons déjà dit, nous pouvons la garder toujours sous contrôle.