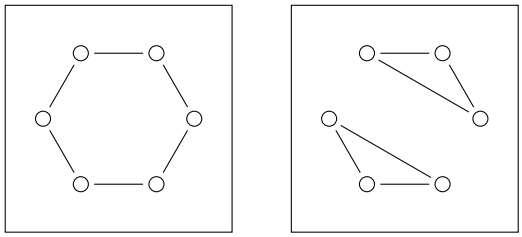
Afstudeerproject (BSc/MSc): Graph Isomorphism and Partition Refinement

Begeleider: Joshua Moerman
joshua.moerman@ou.nl

Samenvatting: Graaf isomorfisme (Engels: *graph isomorphism*) is een belangrijk probleem in de informatica. Bij dit algoritmisch probleem zijn twee grafen gegeven en de vraag is of er een functie bestaat die de ene graaf afbeeldt op de andere graaf, waarbij de lijnen behouden zijn.



Figuur 1: Twee grafen die *niet* isomorf zijn.

Het is nog onbekend of er een efficiënt (dat willen zeggen: *polynomiaal*) algoritme bestaat dat beslist of twee grafen isomorf zijn. Toch zijn er tools die dit probleem vaak snel kunnen oplossen, zoals `nauty` [MP14]. Één van de stappen in het algoritme van `nauty` is het verfijnen van partities (Engels: *partition refinement*). Recentelijk is er een nieuw algoritme voor partitieverfijning [JW23] en de vraag is of graaf isomorfisme ook sneller kan met dit nieuwe algoritme.

Voor een bachelor-project ligt de focus vooral op het vergelijken (benchmarken) van de bestaande tools. Hiervoor zullen wel enkele artefacten geïmplementeerd moeten worden, bijvoorbeeld om de inputs te converteren en de tools te draaien. Voor een master-project zal er daarnaast ook een theoretisch component zijn. Zo kan graaf-isomorfie ingezet worden om isomorfie van hyper-grafen, "designs" en meetkundige objecten aan te tonen. Daarvoor is dan wel een reductie nodig. Aan de andere kant is de `boa` tool generiek in het type coalgebra. Het is nog onduidelijk hoe de partitieverfijning van zulke objecten zich verhoudt tot de grafen die erbij horen.

Keywords: *Graphs, Algorithms, Combinatorics, Complexity Theory*

Achtergrond

Grafentheorie

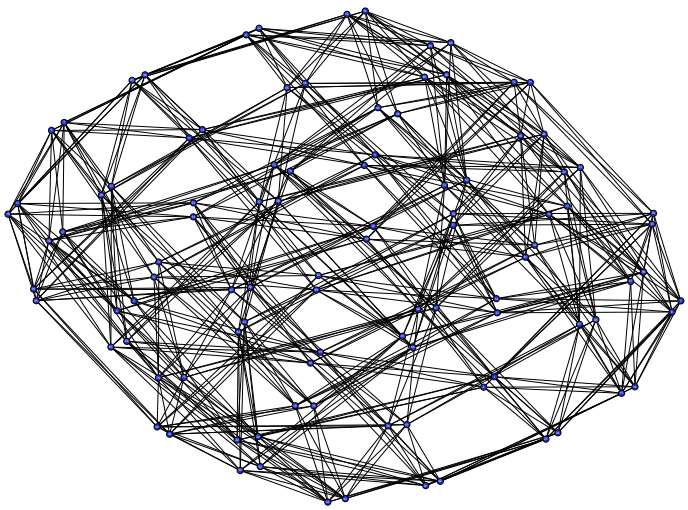
Een graaf G bestaat uit een verzameling punten V (voor **v**ertices) en een verzameling lijnen E (voor **e**dges). Een lijn is een deelverzameling van V met precies twee elementen: het beginpunt van een lijn en een eindpunt.

Als we twee grafen $G_1 = (V_1, E_1)$ en $G_2 = (V_2, E_2)$ hebben, kunnen we ons afvragen of ze "hetzelfde" zijn. De technische term hiervoor is *isomorfie*. Twee grafen zijn isomorf als er een functie $f : V_1 \to V_2$ bestaat zodat:

1. f bijectief is, en
2. voor alle $u, v \in V_1$ geldt $\{u, v\} \in E_1 \iff \{f(u), f(v)\} \in E_2$.

In figuur 1 zien we twee grafen. Sommige aspecten van de grafen zijn gelijk: ze hebben evenveel punten en elk punt heeft dezelfde graad (namelijk 2). Maar andere aspecten zijn verschillend, zo is het aantal componenten duidelijk anders: de linker graaf is samenhangend en de rechter graaf niet. Deze grafen kunnen daarom niet isomorf zijn.

Graaf isomorfie komt in vele vakgebieden voor. Zo wordt het gebruikt bij het vergelijken van moleculen, het verifiëren van electronische circuits, enzovoorts. De complexiteit van het graaf isomorfie probleem is nog niet opgelost: het zit zeker in NP, maar we weten niet of het in P zit én we weten ook niet of het NP-volledig is. Algoritmes die voor de meeste grafen efficiënt werken bestaan wel, zoals de tools `nauty` [MP14], `Traces` [Pip08] en `bliss` [JK12].



Figuur 2: Grafen kunnen ook groot zijn, en dan wordt isomorfie complex.

Partitieverfijning

Een van de subroutines in `nauty` is het zogenaamde partitieverfijnen. Hierbij proberen we de punten van een graaf te kleuren, afhankelijk van de eigenschappen die ze hebben. Punten met verschillende graad krijgen bijvoorbeeld ook een andere kleur. Een isomorfisme zal altijd die kleur moeten behouden, en dus wordt de zoekruimte naar een isomorfisme kleiner als we de kleuren precies kiezen.

Deze methode werd uitgevonden in de context van grafentheorie [WL68], maar ook in de context van automatentheorie [Moo56].

De eerste die een erg efficiënt algoritme vond voor partitieverfijning was Hopcroft [Hop71] en dit werd later aangepast voor grafen [CC82] en gelabelde transitiesystemen [PT87].

Coalgebra

Zoals in de samenvatting aangekondigd is er recentelijk een nieuw algoritme voor partitieverfijning [JW23]. Dit algoritme komt uit de abstracte theorie van *coalgebra*. Dit is een overkoepelende theorie over toestanssystemen (denk aan automaten, toestandsdiagrammen, enzovoorts).

Een graaf kunnen we ook zien als toestandsdiagram. Elk punt is dan een toestand, en de transities leiden naar de punten die verbonden zijn met een lijn. Een graaf is dan niet meer gegeven als (V, E), maar als een functie $V \to \mathcal{P}(V)$, die voor elk punt verteld wat de verbonden punten zijn.

Partitieverfijning is een zeer bekend begrip in de theorie van coalgebra en wordt veel gebruikt bij, bijvoorbeeld, model checking [BK08].

Referenties

Hieronder staan alle genoemde referenties, je hoeft ze uiteraard niet allemaal te lezen. Voor meer informatie over de graaftheoretische kant van partitieverfijning, kun dit je delen van dit proefschrift lezen [Kie20].

[BK08] Christel Baier en Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[CC82] Alain Cardon en Maxime Crochemore. “Partitioning a graph in O (n log n)”. In: *Theoretical Computer Science* 19.1 (1982), p. 85–98.

[Hop71] John Hopcroft. “An n log n algorithm for minimizing states in a finite automaton”. In: *Theory of machines and computations*. Elsevier, 1971, p. 189–196.

[JK12] Tommi Junttila en Petteri Kaski. “bliss: A tool for computing automorphism groups and canonical labelings of graphs”. In: *URL* (2012). http://www.tcs.hut.fi/Software/bliss.

[JW23] Jules Jacobs en Thorsten Wißmann. “Fast Coalgebraic Bisimilarity Minimization”. In: *Proc. ACM Program. Lang.* 7.POPL (2023), p. 1514–1541.

[Kie20] Sandra Kiefer. “Power and Limits of the Weisfeiler-Leman Algorithm”. Proefschrift. RWTH Aachen University, 2020.

[Moo56] Edward F Moore. “Gedanken-experiments on sequential machines”. In: *Automata studies* 34 (1956), p. 129–153.

[MP14] Brendan D. McKay en Adolfo Piperno. “Practical graph isomorphism, II”. In: *J. Symb. Comput.* 60 (2014), p. 94–112.

[Pip08] Adolfo Piperno. “Search Space Contraction in Canonical Labeling of Graphs (Preliminary Version)”. In: *CoRR* abs/0804.4881 (2008).

[PT87] Robert Paige en Robert E Tarjan. “Three partition refinement algorithms”. In: *SIAM Journal on computing* 16.6 (1987), p. 973–989.

[WL68] Boris Weisfeiler en Andrei Leman. “The reduction of a graph to canonical form and the algebra which appears therein”. In: *nti, Series* 2.9 (1968), p. 12–16.