

Algorithms and Networks

Graph Isomorphism

Hans Bodlaender and Stefan Kratsch

March 24, 2011

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Graph Isomorphism

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be graphs.

An isomorphism of G_1 to G_2 is a bijection $\phi : V_1 \rightarrow V_2$ s.t.:

$$\{u, v\} \in E_1 \Leftrightarrow \{\phi(u), \phi(v)\} \in E_2$$

G_1 and G_2 are isomorphic: there exists an isomorphism of G_1 to G_2

injective: $\phi(u) = \phi(v) \Rightarrow u = v$

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

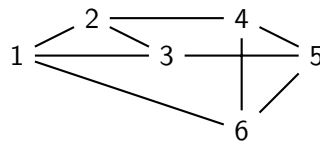
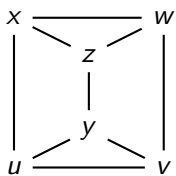
Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Two isomorphic graphs



$$\begin{aligned} \phi(u) &= 6 & \phi(v) &= 5 & \phi(w) &= 3 \\ \phi(x) &= 1 & \phi(y) &= 4 & \phi(z) &= 2 \end{aligned}$$

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



The Graph Isomorphism problem

Graph Isomorphism

Input: Two graphs G_1 and G_2 .

Output: Are G_1 and G_2 isomorphic?

naive algorithm: try all $n!$ injective functions from $V(G_1)$ to $V(G_2)$

natural extension: isomorphism of colored or labeled graphs

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Applications

- ▶ chemistry: check whether two given molecules are the same (e.g., to find a molecule in a database)
- ▶ automated circuit design: does the circuit layout match the initial design?
- ▶ as a subroutine in other algorithms (e.g., maintaining a set of subproblems that were already solved)

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Universiteit Utrecht

5

Outline

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Universiteit Utrecht

6

Outline

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Universiteit Utrecht

7

Complexity of the Graph Isomorphism problem

not known to be in P: no polynomial time algorithm is known
best known worst-case runtime: $\mathcal{O}(c^{\sqrt{n \log n}})$

it is contained in NP: a given isomorphism can be checked in polynomial time

unlikely to be NP-complete: Schoening's *lowness* proof

not known to be in coNP: are there short proofs of non-isomorphism?

Theorem: If Graph Isomorphism is NP-complete then the polynomial hierarchy collapses. [Schoening]

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Universiteit Utrecht

8

Intermediate?

Ladner's Theorem: If $P \neq NP$ then there exist intermediate problems in NP that are neither in P nor NP-complete.

- ▶ problem constructed by Ladner's Theorem is artificial
- ▶ however, many researchers believe Graph Isomorphism to be intermediate
⇒ introduced class of problems that are as hard as GI

A problem is Graph Isomorphism complete if it is equivalent to Graph Isomorphism under polynomial time many-one reductions.



Outline

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Some GI-complete problems

- ▶ isomorphism of hypergraphs
- ▶ isomorphism of colored graphs
- ▶ isomorphism of finite automata
- ▶ checking whether a graph is self-complementary
- ▶ counting the number of isomorphisms between two graphs

in particular: Graph Isomorphism is GI-complete even when restricted to various simple graph classes



GI-complete special cases of Graph Isomorphism

Graph Isomorphism on \mathcal{C} graphs

Input: Two graphs $G_1, G_2 \in \mathcal{C}$.

Output: Are G_1 and G_2 isomorphic?

various graph classes \mathcal{C} with GI-complete isomorphism problem:

- ▶ connected graphs, graphs of minimum degree at least c ,
- ▶ directed acyclic graphs,
- ▶ bipartite graphs, split graphs, chordal graphs,...
- ▶ on $co - \mathcal{C}$ if it is GI-complete on \mathcal{C}

Concept: If there would be a polynomial-time algorithm for, e.g., bipartite graphs, that could be used to efficiently decide isomorphism of general graphs.



Simple hardness proofs

Theorem: Graph Isomorphism of connected graphs is GI-complete.

idea: add a universal vertex to both graphs

Lemma: Two graphs are isomorphic if and only if they are isomorphic after adding a universal vertex to both.

Theorem: Graph Isomorphism of graphs with minimum degree at least c is GI-complete.

idea: add ?? universal vertices to both graphs

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Universiteit Utrecht

13

GI-completeness of GI on bipartite graphs I

Given a graph $G = (V, E)$. Replace each edge $\{u, v\}$ of G by two edges $\{u, x_{u,v}\}$ and $\{x_{u,v}, v\}$ where $x_{u,v}$ is a new vertex.
"Subdividing all edges."

Let $\pi(G)$ denote the obtained graph on $|V| + |E|$ vertices and $2|E|$ edges.

Lemma: Two graphs G_1 and G_2 of minimum degree at least 3 are isomorphic if and only if $\pi(G_1)$ and $\pi(G_2)$ are isomorphic.

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Universiteit Utrecht

14

GI-completeness of GI on bipartite graphs II

Lemma: Two graphs G_1 and G_2 of minimum degree at least 3 are isomorphic if and only if $\pi(G_1)$ and $\pi(G_2)$ are isomorphic.

Theorem: Graph Isomorphism of bipartite graphs is GI-complete.

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Universiteit Utrecht

15

Outline

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Universiteit Utrecht

16

Graph classes with efficient isomorphism testing

- ▶ trees
- ▶ planar graphs
- ▶ interval graphs
- ▶ graphs of bounded genus
- ▶ graphs of bounded degree
- ▶ graphs of bounded treewidth
- ▶ colored graphs such that each vertex has either bounded degree or bounded co-degree into each color
- ▶ ...



Universiteit Utrecht

17

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems

Polynomial time algorithm for tree isomorphism

Given two trees G_1 and G_2 , decide whether they are isomorphic.

basic ideas:

- ▶ every nontrivial tree has (at least two) leaves
- ▶ proceed in rounds: each round remove all leaves
- ▶ use colors to encode the removed leaves



Universiteit Utrecht

18

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems

Polynomial time algorithm for tree isomorphism

first try:

- ▶ (in both given trees) remove all leaves
- ▶ assign each vertex an integer (the color) = number of adjacent leaves that were removed
- ▶ recurse/iterate?

observation:

- ▶ must be able to handle colored input
- ▶ good: two vertices have the same color \Rightarrow lost the same number of leaves (don't need to know how many!)



Universiteit Utrecht

19

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems

Polynomial time algorithm for tree isomorphism

second try:

- ▶ given two colored trees both on n vertices ($\Rightarrow \leq n$ colors)
- ▶ remove all leaves
- ▶ n' = number of remaining vertices (return NO if different)
- ▶ assign each vertex an integer array (a_1, a_2, \dots, a_n)
 a_i = number of adjacent leaves of color i that were removed
- ▶ replace the arrays by numbers $1, \dots, n' < n$:
same array entries \Rightarrow same number
- ▶ repeat, while $n' > 2$
- ▶ if both trees are colored K_1 or K_2 then compare colors
return: YES if same, NO if different



Universiteit Utrecht

20

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems

Outline

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Universiteit Utrecht

21

Introduction
Some facts about Graph Isomorphism
Proving Graph Isomorphism completeness
Efficient algorithm for isomorphism of trees
Iterative refinement heuristic
Open problems

Why heuristics?

- ▶ no algorithm is known that provably decides isomorphism in polynomial time
- ▶ **but:** Graph Isomorphism is easy on “most instances”
- ▶ it often very easy to show that two graphs are not isomorphic:
 - different number of vertices or edges
 - different degree sequences
 - mismatch in any other isomorphism invariant property



Universiteit Utrecht

22

Introduction
Some facts about Graph Isomorphism
Proving Graph Isomorphism completeness
Efficient algorithm for isomorphism of trees
Iterative refinement heuristic
Open problems

Iterative refinement: Central idea

- ▶ for isomorphism ϕ from G_1 to G_2 it must hold that

$$\{u, v\} \in E_1 \Leftrightarrow \{\phi(u), \phi(v)\} \in E_2$$

⇒ in particular u has the same degree as $\deg(u)$

- ▶ thus for any constant c : vertices of degree c in G_1 must be mapped to vertices of degree c in G_2
- ▶ the same is true for the number of neighbors that have degree c ...

of course we don't have ϕ yet, but we will use these properties to find it (if it exists)



Universiteit Utrecht

23

Introduction
Some facts about Graph Isomorphism
Proving Graph Isomorphism completeness
Efficient algorithm for isomorphism of trees
Iterative refinement heuristic
Open problems

Iterative refinement: Central idea (again)

- ▶ partition the vertices of both graphs into classes
- ▶ each class of G_1 has a corresponding class in G_2
- ▶ vertices of any class must be mapped to vertices of the corresponding class
- ▶ refine classes as long as possible
- ▶ finally: check all possible mappings that remain



Universiteit Utrecht

24

Introduction
Some facts about Graph Isomorphism
Proving Graph Isomorphism completeness
Efficient algorithm for isomorphism of trees
Iterative refinement heuristic
Open problems

Iterative refinement: step by step

given $G = (V, E)$ and $G' = (V', E')$

- ▶ return “non-isomorphic” if $|V| \neq |V'|$ or $|E| \neq |E'|$
- ▶ partition the vertices according to degree:

$$V = V_0 \dot{\cup} V_1 \dot{\cup} \dots \dot{\cup} V_{n-2} \dot{\cup} V_{n-1}$$
$$V' = V'_0 \dot{\cup} V'_1 \dot{\cup} \dots \dot{\cup} V'_{n-2} \dot{\cup} V'_{n-1}$$

where $V_c = \{v \in V \mid \deg(v) = c\}$ (ditto for V'_c)

- ▶ clearly $\phi(V_c) = V'_c$ for any isomorphism of G_1 to G_2
- ▶ return “non-isomorphic” if $|V_c| \neq |V'_c|$ for any $c \in \{0, \dots, n-1\}$



Iterative refinement: step by step

now let's refine the sets further:

- ▶ if a vertex $v \in V_c$ has t neighbors in V_c then $\phi(v) \in V'_c$ and it has t neighbors in $V'_c \Rightarrow$ refine the sets V_c according to the number of neighbors in sets V_c (in both graphs)
- ▶ if any two corresponding sets differ in size, then return “non-isomorphic”
- ▶ if any two corresponding sets have size one, then each isomorphism must respect that

keep refining until all vertices from any set have the same number of neighbors in any (other) set



Iterative refinement: other properties

other properties of vertices to refine by

- ▶ number of triangles that contain v (or any other such small graphs)
- ▶ shortest path distances from v (e.g., 5 vertices at distance one, 10 vertices at distance two, ...)
- ▶ <your-suggestion-here>

trade-off: more time investment \Rightarrow possibly smaller sets

does this heuristic work for colored graphs? – certainly!



Iterative refinement: Wrap-up

- ▶ partition the vertex sets of both graphs and refine as much as possible
- ▶ try all possible isomorphisms that map vertices from each set to vertices of the corresponding set in the second graph
- ▶ hopefully save a lot of time as compared to trying all $n!$ possible isomorphisms
- ▶ in practice this is highly successful (most of the time it will quickly discover a proof for non-isomorphism)



Outline

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems



Universiteit Utrecht

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems

Open problems

- ▶ is Graph Isomorphism in P?
- ▶ is Graph Isomorphism in coNP?
- ▶ assuming $P \neq NP$, can one show $GI \notin P$?



Universiteit Utrecht

Introduction

Some facts about Graph Isomorphism

Proving Graph Isomorphism completeness

Efficient algorithm for isomorphism of trees

Iterative refinement heuristic

Open problems