University of St Andrews

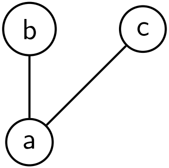
# Between Subgraph Isomorphism and Maximum Common Subgraph, How to make faster algorithms

**Ruth Hoffmann**, Mun See Chang, Ciaran McCreesh and Craig Reilly

Scottish Combinatorics Meeting, Strathclyde University

# Graph

Let $G = (V, E)$ be a graph, with the vertex set $V$ and edge set $E = \{(v, u) : v, u \in V\}$.




$$V = \{a, b, c\}, \; E = \{(a, b), (a, c)\}$$

# Applications

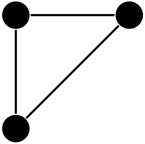
- Networks (Traffic, Public Transport, Computer, Social Media, Social, Disease, etc.)
- Chemical Compounds
- Protein Interactions
- Circuits
- Databases

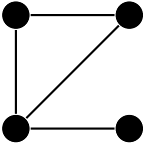
# Finding Patterns

- Find a smaller structure inside a larger ones.
- Find a common structure between two given ones.
- Permutation Patterns?
- Graph Patterns

# Subgraph Isomorphism Problem

Given two graphs $P = (V_P, E_P)$ and $T = (V_T, E_T)$ we try to find the pattern graph $P$ inside the target graph $T$.



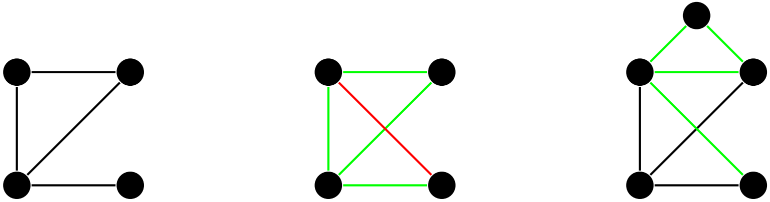


# Non-Induced SIP

Given two graphs $P = (V_P, E_P)$ and $T = (V_T, E_T)$ we ask if there exists an injection $f : V_P \to V_T$ such that $(u, v) \in E_P$ iff $(f(u), f(v)) \in E_T$.

# Induced SIP

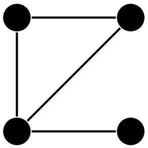
Given two graphs $P = (V_P, E_P)$ and $T = (V_T, E_T)$ we ask if there exists an injection $f : V_P \to V_T$ such that $(u, v) \in E_P$ iff $(f(u), f(v)) \in E_T$ and $(v, u) \notin E_P$ iff $(f(u), f(p)) \notin E_T$.

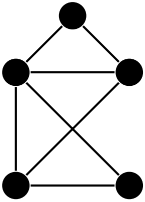


# Computationally

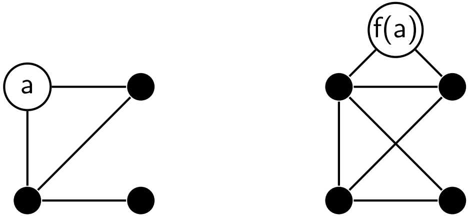
- SIP is NP-Complete
- Algorithms build on backtrack search where we keep matching $v_P \in V_P$ with $v_T \in V_T$ until it breaks $f$ (and then we backtrack) or gives us a solution (when all $v_P$ have been matched without breaking $f$).

# Computationally

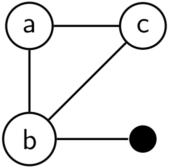


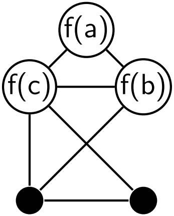


# Computationally

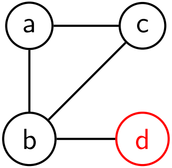


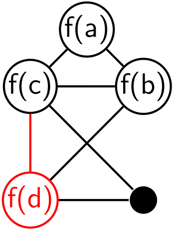
# Computationally



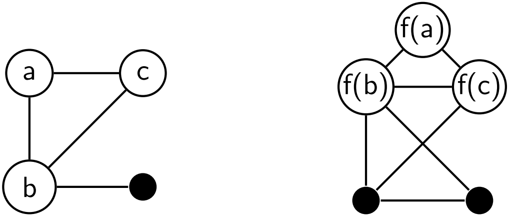



# Computationally

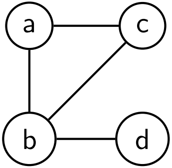


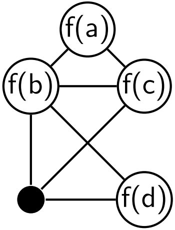



# Computationally



# Computationally



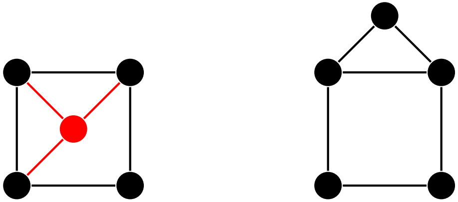
# Improvements

Search can be "improved" (made more efficient) by creating new algorithms which use

- Colours (clique colourings)
- Auxiliary graphs
- Parallelism
- Specialised heuristics
- Graph properties

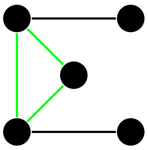
# $k$-less Subgraph Isomorphism

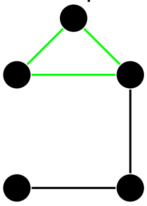
The $k$-less subgraph isomorphism $\kappa : P \rightarrow T$ is a subgraph isomorphism from all but $k$ vertices of $P$ to $T$. Can be induced and non-induced.



# Common Induced Subgraph

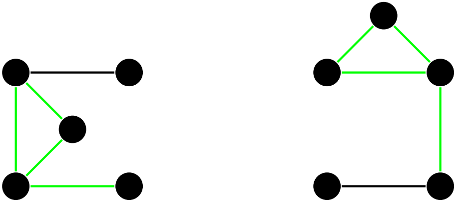
A common induced subgraph of graphs $G$ and $H$ is a graph $P$, such that there is an induced subgraph isomorphism from $P$ to $G$ and an induced subgraph isomorphism $P$ to $H$.





# Maximum Common Induced Subgraph

The common induced subgraph of graphs $G$ and $H$ is a graph $P$, such that there is an induced subgraph isomorphism from $P$ to $G$ and an induced subgraph isomorphism $P$ to $H$ such that there is no $P'$ which is a common induced subgraph of $G$ and $H$ with $|V_{P'}| > |V_P|$.



# $k$-less in action in MCS

$k$-less SIP is essentially asking to find a common subgraph between $P$ and $T$ with $|V_P| - k$ vertices.

- Filtering using degrees
- Filtering during search using paths

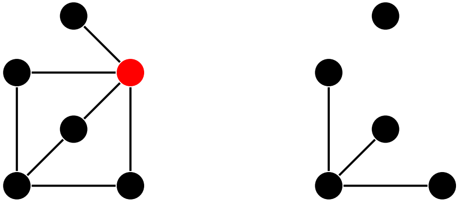
# Degree filtering

- Let $p$ be a vertex in $P$ and $t$ a vertex in $T$. For both non-induced and induced $k$-less subgraph isomorphisms, if $p \to t$ then $deg(p) - k \leq deg(t)$.
- If $p \to t$ then the neighbourhood degree sequence of $p$ is less than or equal to the neighbourhood degree sequence of $t$.

We compare two sequences $S$ and $T$ and say that $S$ is smaller than $T$ is $S$ is shorter or for all entries in $S$ there exists a distinct entry in $T$ which is greater or equal to it.

# Path Filtering

Let $p, q \in V_P$ and $t, u \in V_T$. If there is a $k$-less isomorphism in which $p \to t$ and $q \to u$ then $\text{paths}(p, q, 2) - k \leq \text{paths}(t, u, 2)$.



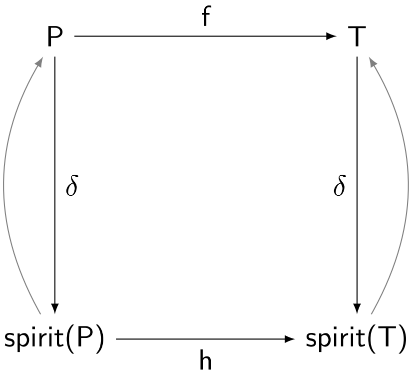
# No new algorithms

What about not coming up with new fancy algorithms? Can we make what is out there better with simple tweaks?

- What can we learn during search?
- Where are solutions likely to be?
- Can we guide search away (intelligently) from places the solution will not be?

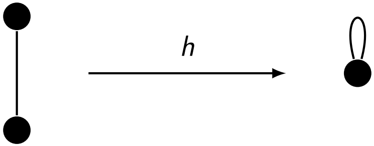
We look at non-induced SIP (we know that one works).

# Distilling & Learning



# Graph Homomorphism

A graph homomorphism $h$ from $G = (V_G, E_G)$ to $H = (V_H, E_H)$ is a mapping $h : V_G \to V_H$ such that if $(u, v) \in E_G$ then $(h(u), h(v)) \in E_H$.



# Distillation

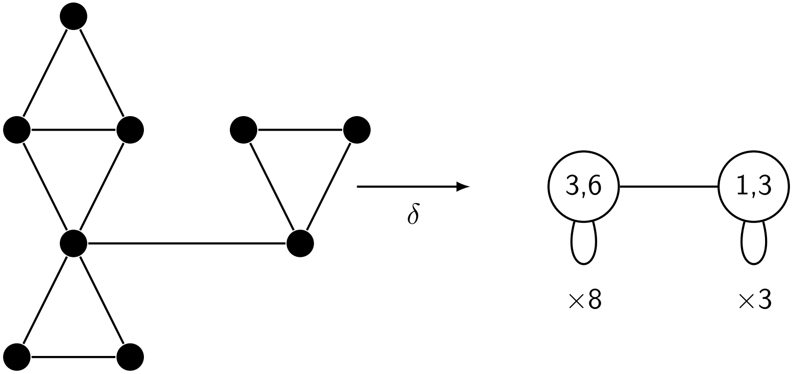
We define $\delta(G)$ to be the quotient of $G$ over the relation $R$ where $u, v \in V_G$ are related ($u \sim_R v$) if there exists a chain $(\Delta_1, \ldots, \Delta_k)$ (a chain of triangles of vertices in $G$) such that $u \in \Delta_1$ and $v \in \Delta_k$ and $\forall i, \exists j \leq i$ such that $\Delta_i \cap \Delta_j \neq \emptyset$.

**Theorem**

*$\delta$ is a graph homomorphism of $G$.*

We call $\delta$ a distillation and $\delta(G) = \text{spirit}(G)$ the spirit of $G$.

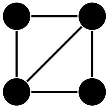
# Distillation Example

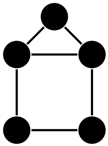


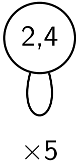
# Using Distillations to help SIP

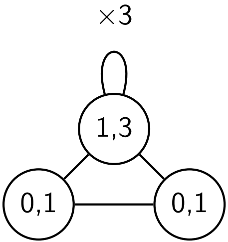
If there exists a subgraph isomorphism $f$ from a pattern graph $P$ to a target graph $T$ then there exists a homomorphism $h$ from spirit($P$) to spirit($T$).

# Examples

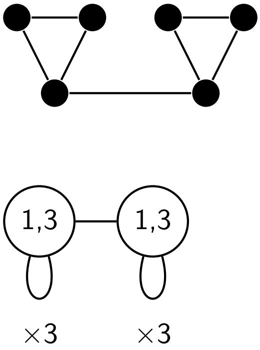


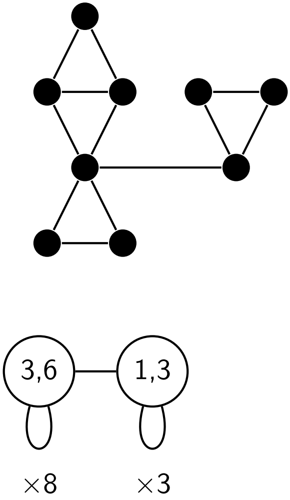




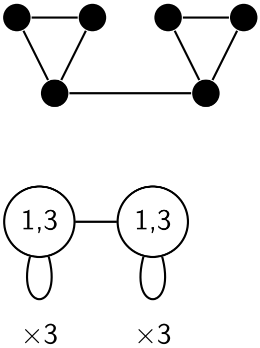



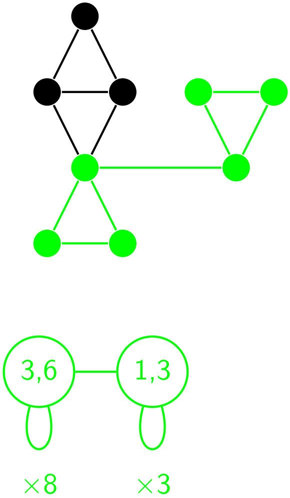
# Examples



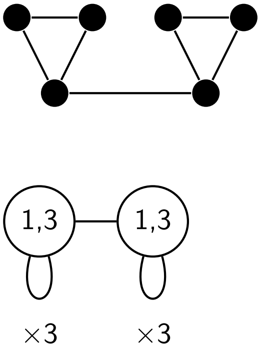



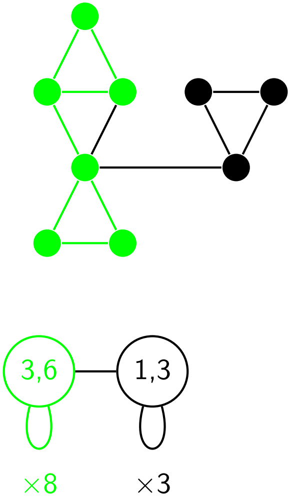
# Examples



# Examples



# Will it work?

- The SIP/Homomorphism problem are (in essence) constraint satisfaction problems
- Everything we learn we can add as constraints
- Even from distilling
- And we can learn from distilling, because we know the distillation is structure preserving

# Will it go faster?

- For no solutions, (hopefully) definitely
- For a solution, maybe
- For many solutions, (hopefully) yes

Thank you!

ruthhoffmann

@ruthhoffmann

rh347@st-andrews.ac.uk