Common Induced Subgraph Isomorphism

Structural Parameterizations and Exact Algorithms

Faisal N. Abu-Khzam

Department of Computer Science and Mathematics
Lebanese American University
Beirut, Lebanon

Overview

- Background material
- (Common) Induced Subgraph Isomorphism
- Complexity on general graphs
- Exact algorithms
- Special graph classes
- Structural parameters
- The size of vertex cover as parameter
- Open problems and Future Directions

Parameterized Problems

- **Parameterized problem**: a instance of a parameterized problem consists of a pair (*I*,*K*) where:
  - *I* is the input
  - *K* is a (set of) parameter(s)
- A parameterized problem is often denoted by (*X*,*k*) or just *k*-*X* (such as *k*-*Vertex Cover*, *k*-*Clique*, etc…)

Fixed-Parameter Tractability

- A problem X is fixed-parameter tractable (**FPT**) with respect to parameter k, if:

  there is an algorithm that solves X in time $O(f(k)n^c)$, where n is the size of input and c is a constant.

Example: the Parameterized Vertex Cover Problem

- A vertex cover in a graph is a set of vertices whose deletion results in an edge-less subgraph
- *k*-Vertex Cover:
  - Given a graph *G*
  - Does *G* have a vertex cover of size *k*?

Example: the Parameterized Vertex Cover Problem

- *k*-Vertex Cover is solvable in O*(2^k):
  - Pick and edge *uv*
  - Either *u* or *v* is in any vertex cover (in each case the vertex is deleted)
  - Search-tree is of height bounded above by *k*

- Better algorithm:
  - pick a vertex *v* of maximum degree
  - Either *v* is placed in cover, or *N*(*v*) is in cover

- Based on this simple idea and other preprocessing/ pruning methods:
  - *k*-Vertex Cover is solvable in $O(1.2745^k k^4 + kn)$ time [Chen et al., 2010]

Kernelization

- For a parameterized problem (X,k), a kernelization algorithm is a **polynomial-time reduction procedure** that takes an arbitrary instance (I,k) of X and produces an equivalent instance (I',k') where $|I'| \leq g(k)$ and $k' \leq k$

- When the above holds, A resulting reduced instance is called a ***g*(*k*)-*kernel***

- Of special interest are kernels where $g(k)$ is a polynomial

k-Vertex Cover Kernelization

- Observe (based on [Buss-Goldsmith, 1993]): every vertex of degree > k must be in any solution. Otherwise we have a no-instance
- Pre-process the graph to delete all the vertices of degree > k and decrement k by the number of such vertices.
  – Repeat until each and every vertex is of degree ≤ k
- The number of edges is now bounded above by k^2
- It follows that k-Vertex Cover admits a quadratic-size kernel
- Admits a kernel with at most $2k$ vertices [Chen-Kanj-Jia, 2001]

The Parameterized Complexity Hierarchy

- The class FPT is at the bottom of the parameterized complexity hierarchy

FPT, W[1], W[2], … **XP**

- The class XP
  - Consists of parameterized problems that are solvable in polynomial time when the parameter is a constant
  - Example: Dominating Set
  - k-Coloring, parameterized by k, is not in XP (unless P=NP)

- FPT versus Kernelization
  - A problem is FPT if and only if it admits a kernel [Downey-Fellows-Stege, 1999]. However:
  - FPT does not imply poly-kernel

Feedback Vertex Set

- A feedback vertex set in a graph is a set of vertices whose deletion results in an acyclic subgraph

- The corresponding *k*-Feedback Vertex Set problem (FVS) is another well known FPT problem
  - $O^*(3.168^k)$ [Kociumaka-Pilipczuk, 2014]
  - Quadratic-size kernel [Thomasse', 2010]

k-Clique

- Clique: complete (sub)graph
  - Any two vertices of a clique are adjacent.

- *k*-Clique is W[1]-hard

- In terms of worst-case behavior, the best known exact algorithm runs in $O(1.213^n)$ [Bourgeois et al., 2010]

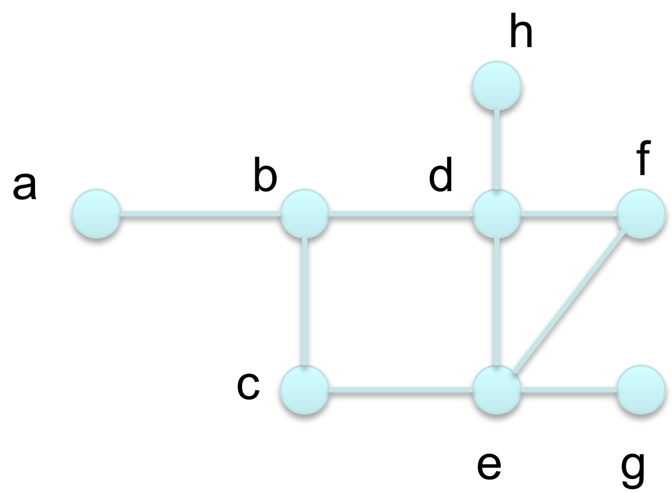
Treewidth

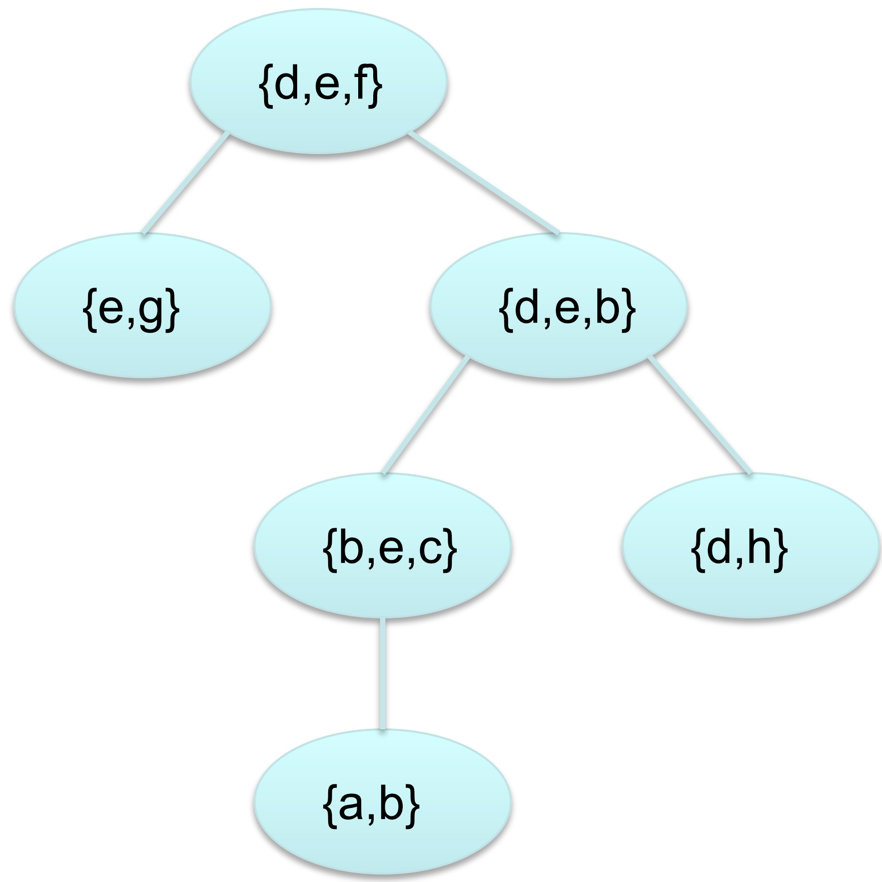
- A tree decomposition of $G=(V,E)$ is a tree $T=(X,Y)$ such that elements of X are subsets of V and:
  – For each vertex u in V, the nodes of T that contain u form a subtree denoted by $T(u)$ (every vertex of V can be mapped to a distinct subtree)
  – Every pair of adjacent vertices are mapped to intersecting subtrees of T

- The width of a tree decomposition is one less the maximum tree-node cardinality (as subset of V)

- The treewidth of G, henceforth ***tw*(*G*)**, is the minimum width among all possible tree decompositions of G

Treewidth (an example)





e2

Pathwidth

- Path decomposition:
  - Same definition as tree decomposition with tree replaced by path
- The corresponding minimum width is called the pathwidth of the graph (denoted ***pw*(*G*)**)
- Obviously: $tw(G) \leq pw(G)$

Why Treewidth?

- Treewidth measures how tree-like a graph is
- Many known NP-hard problems are in P on graphs of bounded treewidth
- *k*-Treewidth is FPT
- While the fixed-parameter algorithm for Treewidth is too slow, approximation algorithms exist that are more efficient and serve most practical purposes

Relation between vc and tw/pw

- If a graph G has a vertex cover C of size k then $pw(G) \leq k$
  – Let P be a path of length n-k
  – Map every vertex of C to P
  – Map every vertex not in C to a unique (distinct) vertex of P, so every vertex of P is the image of exactly k+1 vertices of G

- It follows that $tw(G) \leq pw(G) \leq vc(G)$

Relation between *fvs*, *vc* and *tw*

- If a graph G has a feedback vertex set S of size k then $tw(G) \leq k+1$
  - Let T be a tree decomposition of G-S
  - Then width(T) = 1 (G-S is a forest)
  - Map every vertex of S to every node of T

- It follows that $tw(G) \leq fvs(G)+1$

- Also note that $fvs(G) < vc(G)$

- Hence $tw(G) \leq fvs(G)+1 \leq vc(G)$

Induced Subgraph Isomorphism (**ISI**)

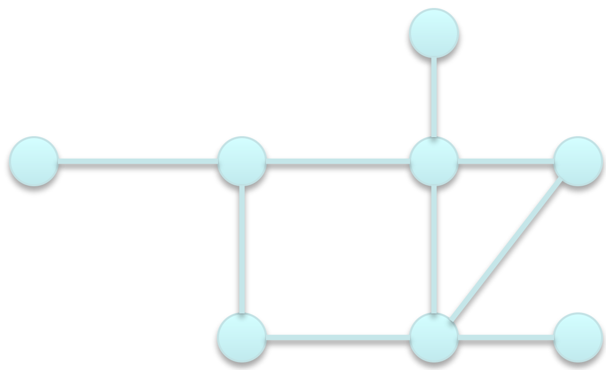
- **Given:** a pair (G_1, G_2) of graphs
- **Parameter:** $|G_1|$
- **Question:** Is G_1 isomorphic to an induced subgraph of G_2?
  - G_1 and G_2 are often called the pattern and host, respectively

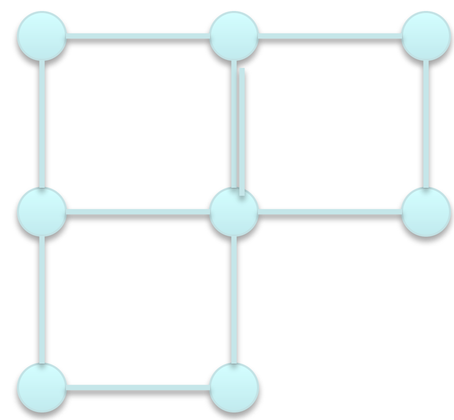
- W[1]-hard in general, by reduction from *k-Clique*
- Fixed-Parameter Tractable in *H*-minor free graphs [Flum-Grohe, 2001]

Maximum Common Induced Subgraph (***MCIS***)

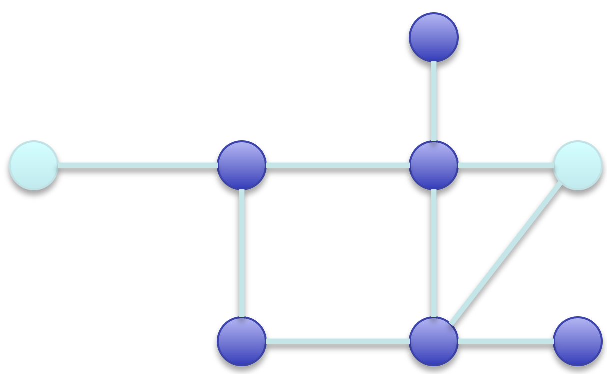
- **Given:** a pair $(G_1, G_{\mathbf{2}})$ of graphs and a positive integer k
- **Question:** Is there a graph H that satisfies:
  – H is isomorphic to an **induced** subgraph of both G_1 and G_2
  – H has at least k vertices
- Other definitions seek:
  – Maximum number of edges
  – Connected common subgraph (henceforth ***MCCIS***)

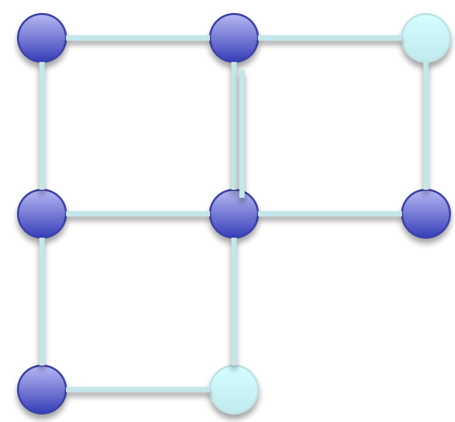
MCIS





MCIS





Complexity of MCIS

- Induced Subgraph Isomorphism is a special case (let *k* = order of pattern graph) :
  - Thus MCIS is NP-hard
- *k-Clique* is another special case.
- MCIS remains NP-hard on most known graph classes
  - Including bipartite graphs, planar graphs, and graphs of bounded treewidth!
- Solvable in polynomial-time on:
  - Trees [Garey & Johnson, 1979]
  - Graphs of bounded treewidth and bounded degree [Akutsu, 1993]

Differentiating between the complexities of ISI, MCIS and MCCIS

- MCIS is hard when the second input graph is edge-less (being equivalent to the Maximum Independent Set problem), while ISI is trivially in P in this case.
- MCCIS is solvable in polynomial time on trees and forests while MCIS is NP-hard in this case.

Exact Algorithms

A Classical Backtracking Algorithm

- (Based on [Ullman, 1976]:) For v in G_1, define $M(v)$ as: set of possible matches of v in G_2
- Initialize $M(v)$ to $V(G_2)$ for each v in G_1.
- MCIS(G_1,G_2,M,k)
  - If $k = 0$ then return YES
  - For every v in $V(G_1)$ do
    - If $M(v)$ is empty, then delete v
  - If G_1 is empty then return NO
  - $M' \leftarrow M$
  - Pick v of G_1 with minimum $|M'(v)|$
  - For each w in $M'(v)$ do:
    - Match v and w:
      - if x is a neighbor of v, delete the non-neighbors of w from M'(x)
      - if x is not a neighbor of v, delete the neighbors of w from M'(x)
    - If(MCIS(G_1-v,G_2-w,M',k-1)) then return YES
  - return MCIS(G_1-v,G_2,M,k)

Analysis

- Let n and m be the number of vertices in G_1 and G_2, respectively.
- In the worst-case, every vertex of G_2 is a possible match ➔ m+1 choices for each vertex of G_1

$$O((m+1)^n)$$

- Of course, the actual bound is better!
- How can we do better?

Reduction to Maximum Clique

- (Based on [Levi-Calcolo, 1972]:) Given an MCIS instance (G_1, G_2, k), construct a Clique instance (H, k) as follows:
  - $V(H) = \{(u,v)$: u and v are vertices of G_1 and G_2, respectively$\}$
  - $E(H) = \{\{(u,v),(u',v')\}$: uu' and vv' exhibit the same relation$\}$

- A clique in H gives rise to a common subgraph!

- Unfortunately, this does not help a lot!
  - The running time would be in $O(1.213^{n^2})$

Reduction to Graph Isomorphism

- For each pair of subgraphs H_1 and H_2 of G_1 and G_2, respectively
  - Run a Graph Isomorphism algorithm on H_1 and H_2

- Running time: $2^{n+m+O((\log n)^c)}$, using Babai's recent algorithm for Graph Isomorphism.

Parameterized Complexity of ISI and M(C)CIS

Parameterized Complexity of ISI and M(C)CIS

- ISI and MCIS have the same parameterized complexity

- In fact, When ISI is FPT, MCIS and MCCIS are FPT:
  – Generate all possible graphs on *k* vertices and run ISI's FPT algorithm

- When MCIS is FPT, ISI and MCCIS must be FPT…

- When MCCIS is FPT, MCIS "might" be FPT:
  – Add a universal (or star) vertex to each of the input graphs
    Caution: this is not always possible

Parameterized Complexity of ISI and M(C)CIS

- **Theorem**: ISI, MCIS and MCCIS are W[1]-**Complete**
- Membership in W[1]:
  - a problem is in W[1] if it can be reduced in FPT-time to simulating a non-deterministic single-tape Turing Machine that halts in $f(k)$ steps, for some f.
- This is obviously true for MCIS:
  - Guess in $2k$ steps the corresponding k vertices of G_1 and k vertices of G_2.
- W[1]-hardness is (again) due the reduction from Clique

Special Graph Classes

H-Minor-Free graphs

- Definition: graph *H* is a minor of graph *G* if *H* is obtained from a subgraph of *G* by a sequence of (zero or more) edge contractions.
- *H*-Minor-Free Graphs: family of finite graphs that exclude a fixed subgraph in the minor order.
- ISI is FPT on *H*-Minor-Free graphs [Flum-Grohe, 2001].
- Thus MCIS and MCCIS are FPT in this case (by previous observation).

Planar graphs

- MCIS is FPT in this case.
- Previous observations give an $O^*(c^{k^2})$ algorithm:
  - Generate all graphs of order k and run ISI twice.
- A simple $O^*(c^k)$ algorithm:
  - If both G_1 and G_2 have > $4k$ vertices then we have a Yes instance
  - Otherwise, one of the two graphs, say G_1, has < $4k$ vertices
  - Enumerate, in $O(2^{4k})$, all induced subgraphs of G_1. For each such subgraph, run the (single-exponential) ISI algorithm of [Dorn 2010

(For the search version, one would compute a *5*-coloring of G_1 and G_2)

- Above method can be used in general on graphs of bounded chromatic number provided ISI is solvable in $2^{o(k^2)}$ (or better).

Trees and Forests

- MCCIS is solvable in poly-time on trees [Gary-Johnson, 79]
- It follows that MCCIS is solvable in polynomial-time on forests:
  - for any pair of trees, one from each of the two input graphs, run a Maximum-Common-Subtree algorithm
- ISI and (therefore) MCIS are NP-hard on forests [Gary-Johnson, 79]
- In general, MCIS is NP-hard on trees
- All three problems are FPT on trees and forests (why?)

Graphs of bounded treewidth

- ISI and MCIS are NP-hard on graphs of bounded treewidth, being already NP-hard on forests.

- Theorem. MCCIS is NP-hard on graphs of treewidth-two.

  Proof: by reduction from Sub-Forest Isomorphism:
  - Given two forests $F1$ and $F2$
  - Add a universal vertex to each forest to obtain G_1 and G_2
  - The resulting graphs have treewidth two
  - Obviously: F_1 and F_2 have a common subgraph of order k if and only if G_1 and G_2 have a common connected subgraph of order $k+1$

Bipartite Graphs

- The NP-hardness of ISI and MCIS follows easily from their NP-hardness on forests
- Induced Matching is W[1]-hard on Bipartite graphs [Moser-Sikdar, 2009]
- Thus ISI and MCIS are both W[1]-hard on Bipartite graphs
- How about MCCIS?

MCCIS on Bipartite Graphs

- Theorem. MCCIS is NP-hard and W[1]-hard on bipartite graphs.

  Proof. By reduction from MCIS. Let (G_1, G_2, k) be an instance of MCIS. For each graph $G_i = (A_i \cup B_i, E_i)$ we construct a bipartite graph $G_i' = (A_i' \cup B_i', E_i')$ as follows
  - $A_i' = A_i \cup \{u_i\}$
  - $B_i' = B_i \cup \{v_i\}$
  - $E_i' = E \cup \{u_i v_i\} \cup \{u_i x : x \text{ in } B\} \cup \{v_i y : y \text{ in } A\}$

- Obviously, a common subgraph of size k in G_1 and G_2 gives rise to a connected common subgraph of size k+2 in G_1' and G_2', and vice versa.

Graphs of Bounded Degree

- ISI is NP-hard when the pattern is a path and the host is a planar cubic graph [Gary-Johnson,79]
- So both MCIS and MCCIS are NP-hard in this case
- ISI is FPT on graphs of bounded degree [Cai et al., 2006]
- It follows that MC(C)IS is in FPT on graphs of bounded degree.

Graphs of bounded degeneracy

- A graph is d-degenerate if the minimum degree of any induced subgraph is $\leq d$
- 1-degenerate graphs are trees & forests
- Thus MCIS is NP-hard on 1-degenerate graphs while MCCIS is in P in this case.
- How about the case $d \geq 2$

Two-degenerate graphs

- Theorem [AbuKhzam-Bonnet-Sikora, 2017]. ISI is W[1]-hard on 2-degenerate graphs.

- Proof. By reduction from clique. Let (G,k) be a Clique instance
  - Construct $(G_1, G_2, k' = k+k(k-1)/2)$ as follows:
  - G_1 is obtained from a k-clique by subdividing each edge once
  - G_2 is obtained from G by subdividing each edge once
  - G_1 is an induced subgraph of G_2 if and only if G contains a k-clique

- It follows that ISI, MCIS and MCCIS are W[1]-hard on d-degenerate graphs for $d \geq 2$

- Corollary. ISI, MCIS and MCCIS are W[1]-hard on **girth-six bipartite** graphs

Interval Graphs

- ISI is NP-Complete and W[1]-hard on interval graphs [Marx-Schlotter, 2010]
- Thus MCIS is also W[1]-hard in this case
- Connected-ISI is solvable in polynomial-time on proper interval graph and bipartite permutation graphs [Heggernes et., 2010]
- How about MCCIS on (proper) Interval Graphs?

Other Graph Classes

- Chordal and bipartite chordal
  - ISI, MCIS are NP-hard. Why?
  - How about MCCIS?
- Cographs
  - ISI and MCIS are NP-hard [Damaschke, 1991]
  - How about MCCIS?
- ISI is solvable in poly-time on
  - 2-connected outerplanar graphs [Syslo, 1982]
  - Graphs of bounded degree and bounded treewidth [Akutsu, 1993]

Structural Parameters

Structural Parameters

- Instead of studying a problem on graphs of bounded treewidth, we may consider using treewidth as parameter.

- This is not the same!

- Other commonly used parameters are: vertex cover and feedback vertex set

- Recall that: for any graph G,

$$\boldsymbol{tw(G) \leq fvs(G)+1 \leq vc(G)}$$

MCIS parameterized by feedback vertex set

- MCIS, parameterized by feedback vertex set, is not in XP (unless P = NP)
  - Proof: simply, MCIS is NP-hard on forests
- MCCIS, parameterized by feedback vertex set, is not in XP (unless P = NP)
  - Proof: same reason as above! Why?
- Corollary: M(C)CIS, parameterized by treewidth, is not in XP

Vertex Cover of only one graph as parameter

- Unless P = NP, ISI is not in XP on graphs where the pattern (only) has a *k*-vertex cover
  - Proof: the case *k*=0 is NP-hard via simple reduction from Independent Set (when the pattern is edgeless)
- So MCIS is not in XP when the parameter is the size of a vertex cover of only one of the input graphs

MCS Parameterized by Vertex Cover

- **Given:** Two graphs G_1 & G_2
- **Parameter:** k = bound on the vertex covers of G_1 & G_2
- **Find:** a graph H of maximum order that satisfies:

  H is isomorphic to an induced subgraph of both G_1 and G_2

- **Theorem: MCIS, parameterized by vertex cover is FPT**

We may also assume vertex covers are given (why?)

MCS Parameterized by Vertex Cover

- Let $n = min(n_1,n_2)$
- Then we have a common subgraph of size n-k
  - Just take the two complements of C_1 & C_2
- The objective is to find a maximum common induced subgraph of order > n-k

Key Lemma

- Lemma [AbuKhzam, 2014]:

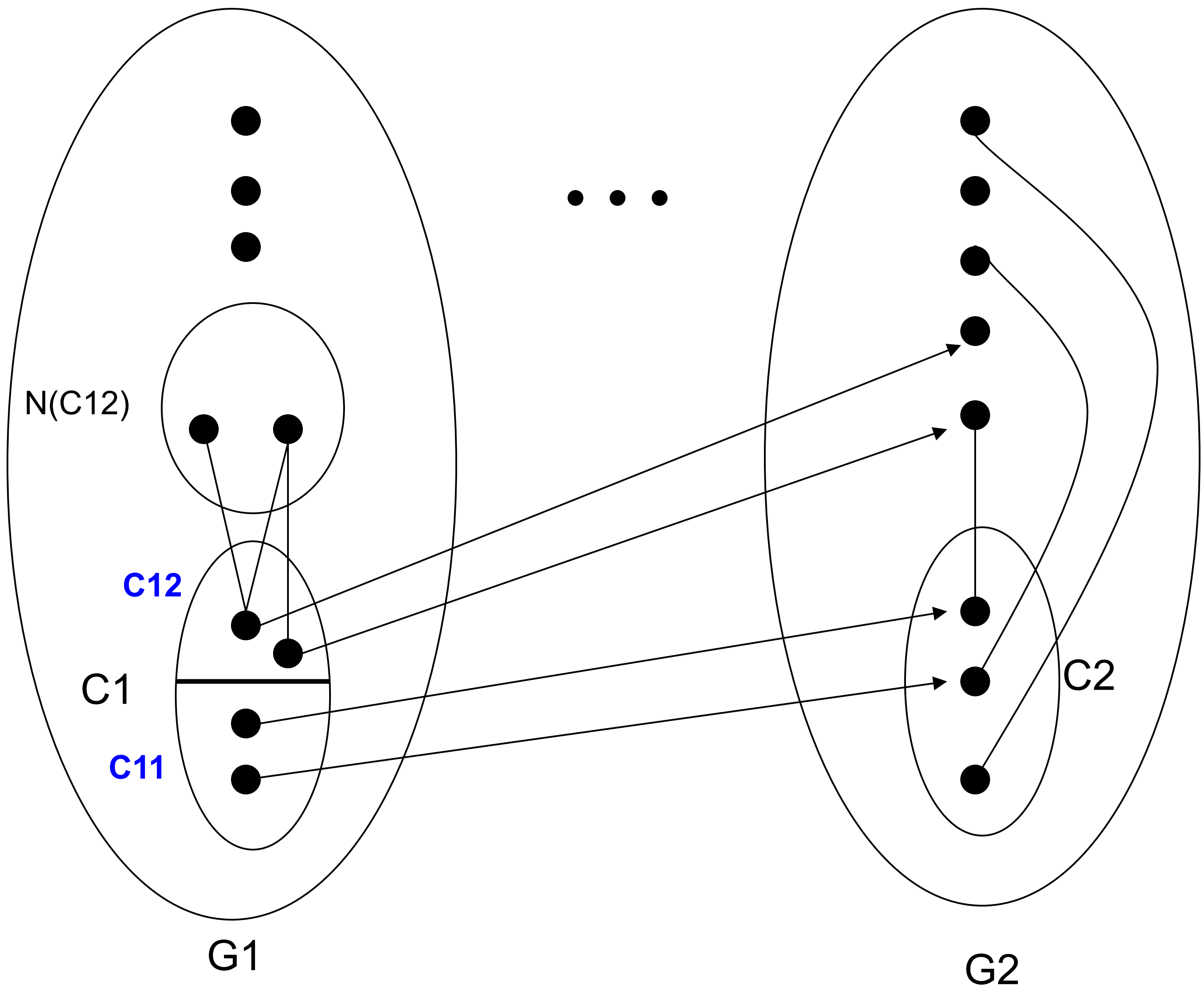
- Let
  - C_{11} = set of elements of C_1 that are matched with elements of C_2.
  - C_{12} = set of elements of C_1 that are matched with elements of I_2 .

- Then: $|N(C_{12})| \leq 2k$

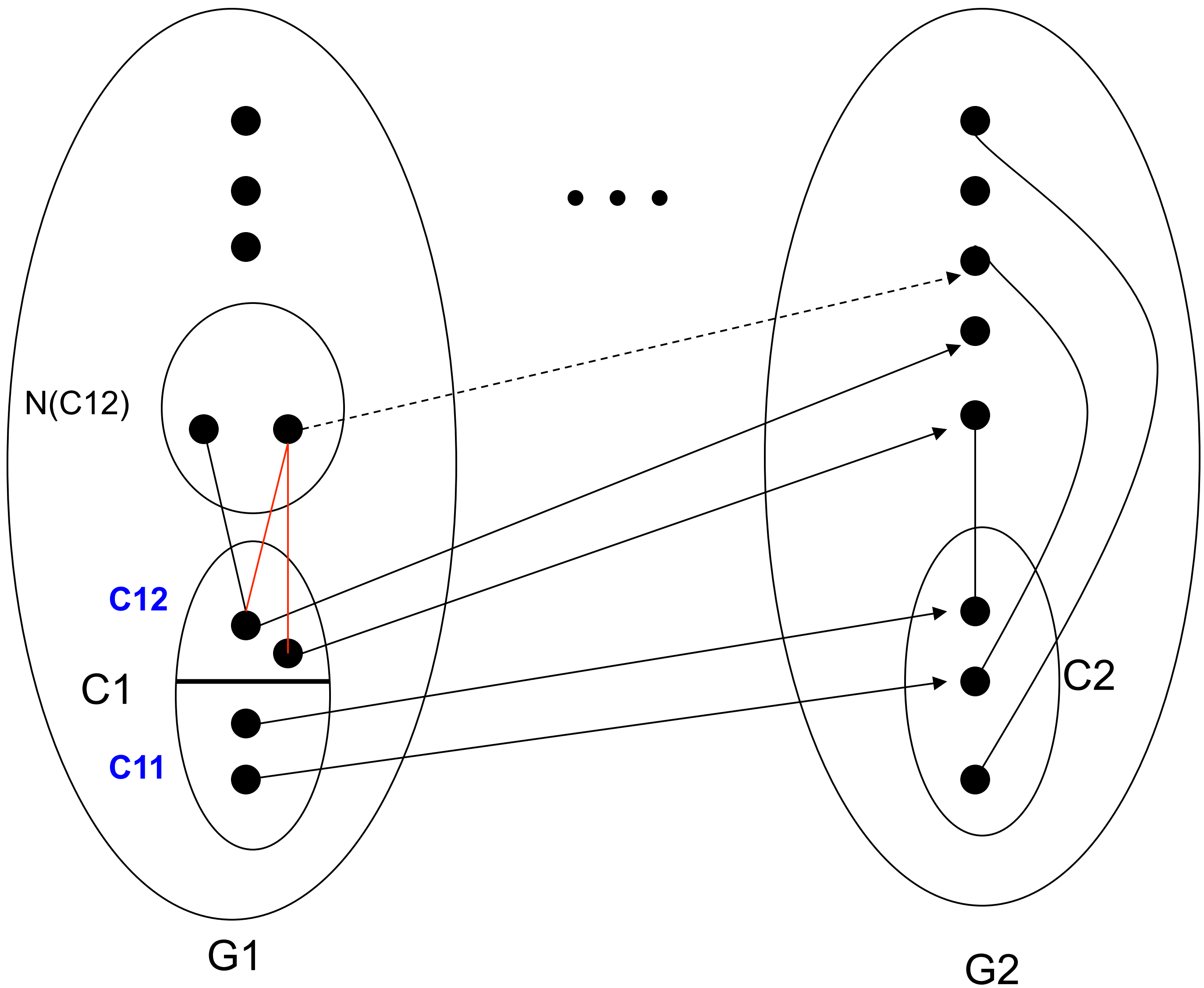
Lemma

- Let
  - C_{11} = set of elements of C_1 that are matched with elements of C_2.
  - C_{12} = set of elements of C_1 that are matched with elements of I_2 .
- Then: $|N(C_{12})| \leq 2k$

- Proof:
  - Elements of $N(C_{12})$ cannot match with any element of I_2 Otherwise… (see figure)
  - If $|N(C_{12})| > 2k$, then at least k elements of $N(C_{i2})$ are unmatched, which makes the independent set solution maximum!



N(C12)
C12
C1
C11
G1
C2
G2



N(C12)
C12
C1
C11
G1
C2
G2

Lemma

- Let
  - C_{11} = set of elements of C_1 that are matched with elements of C_2.
  - C_{12} = set of elements of C_1 that are matched with elements of I_2 .
- Then: $|N(C_{12})| \leq 2k$

- Proof:
  - Elements of $N(C_{12})$ cannot match with any element of I_2 Otherwise… (see figure)
  - If $|N(C_{12})| > 2k$, then at least k elements of $N(C_{12})$ are unmatched, which makes the independent set solution (of size $n-k$) maximum!

A Charge and Reduce Algorithm

Step 1. Branch on elements of C_1 as follows:

- Pick v from C_1
  - Either v is matched with an element of C_2
  - Or v is matched with an (unknown yet) element of I_2
  - Or v is unmatched

A Charge and Reduce Algorithm

- Step2: Branch on elements of $N(C_{12})$
  - Pick a vertex v of $N(C_{12})$:
    - either v matches with an element of C_2
    - or v is unmatched

A Charge and Reduce Algorithm

- Step3: Branch on remaining elements of C_2
  - Such elements must either match to elements of I_1 or to none. So they must form an independent set in G_2
  - For each independent subset C_{22} of C_2 do
    - Delete all neighbors of C_{22} in G_2 (why?)
    - Build a compatibility graph H and proceed by solving Maximum Matching (see next slide)

Matching Vertices via Graph Matching

- After branching on all the elements of C_1 and C_2, we are left with independent sets in each of the two graphs
- We build a bipartite graph $H = (A,B)$ as follows:
  - A and B consists of the unmatched and undeleted elements of G_1 and G_2 respectively
  - Two elements x and y of A and B (resp) are adjacent if their matching does not violate the isomorphism criterion
- Therefore: a maximum matching of H gives the maximum number of pairs of vertices that can match under a common subgraph isomorphism
- The algorithm ends by computing a maximum matching in H and comparing the total number of matched vertices to n-k

Better Branching

- Delay Assignments of elements of C_1 to elements of C_2.
  - Each vertex is either
    - matched (but unassigned a match) or
    - belongs to C_{12} or
    - deleted
  - This would still lead to computing $N(C_{12})$
  - Then branch on elements of $N(C_{12})$ (either matched or deleted)
  - Then try all possible k^k matchings between $C_{12} \cup N(C_{12})$ and C_2
- Total running time is in $O^*((24k)^k)$

Can we do better?

- The main question at this stage is whether an $O^*(c^k)$ algorithm exists when k is the vertex cover bound. Unfortunately:

- **Theorem** [Abukhzam-Bonnet-Sikora, 2017]. Unless the Exponential-Time Hypothesis (ETH) fails, ISI cannot be solved in $O(2^{o(k \log k)})$

MC**C**IS Parameterized by Vertex Cover

- Theorem [AbuKhzam-Bonnet-Sikora, 2017]: MCCIS, parameterized by vertex cover is FPT

  Proof-sketch: Let C_1 and C_2 be vertex covers of G_1 and G_2 resp.,

  - Key observation: elements of the complement of each cover can be partitioned into at most 2^k "twin classes"
  - Enumerate all tri-partitions of C_1 and C_2 into (i) vertices that are matched within covers, (ii) vertices that are matched with other elements and (iii) unmatched vertices.
  - Proceed by enumerating all possible matches between each cover and the complement of the other, then between all the twin-classes

- Also works as MCIS enumeration

Hardness of Kernelization w.r.t. the Vertex Cover Parameter

- Another important question is whether MC(C)IS, parameterized by vertex cover, admits a polynomial-size kernel. Unfortunately:

- Theorem [AbuKhzam-Bonnet-Sikora, 2014]: unless NP is contained in co-NP/poly, MC(C)IS has no polynomial-size kernel when parameterized by the sum of the sizes of vertex covers of the two input graphs.

Some research directions

- We showed MCIS is FPT when parameterized by size of the largest among the vertex covers of input graphs, or just their sum (vc+vc)

- How about the parameter vc+fvs?
  sum of sizes of vertex cover of one graph and feedback vertex set of the other?

- It would be interesting to consider other known graph metrics as parameters:
  - cutwidth, pathwidth, rankwidth, etc ..

- MCIS is solvable in polynomial time on graphs of bounded degree and bounded treewidth
  - How about bounded degeneracy and bounded treewidth?

- How about bipartite graphs of bounded treewidth? Or bounded (given) chromatic number and bounded treewidth?

Thank You