Fragments of domination theory in graphs: invariants, interrelations, and algorithmic aspects (I)

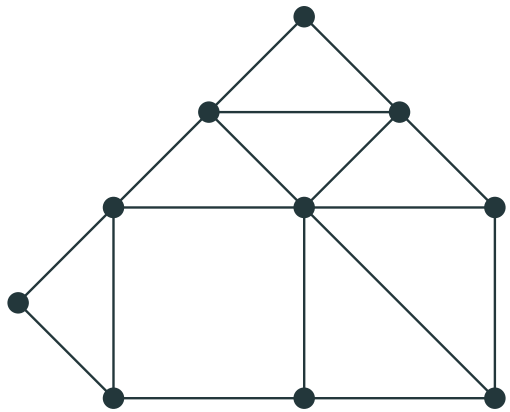
Martin Milanič, University of Primorska
martin.milanic@upr.si
CIMPA Research School on Graph Structure and Complex Network Analysis
Nesin Mathematical Village, Turkey

June 11, 2023

Outline of the lectures:

1. Motivations and background
2. Efficient algorithms for special cases – three examples
3. Approximating the minimum dominating set problem via Set Cover
4. Interrelations between graph domination parameters
5. Approximating the vector dominating set via Submodular Cover

In this series of lectures, all the graphs will be **finite, simple, and undirected**.



Notation: $G = (V, E)$

V is the **vertex set**

E is the **edge set**

Why don't we care about vertex names?

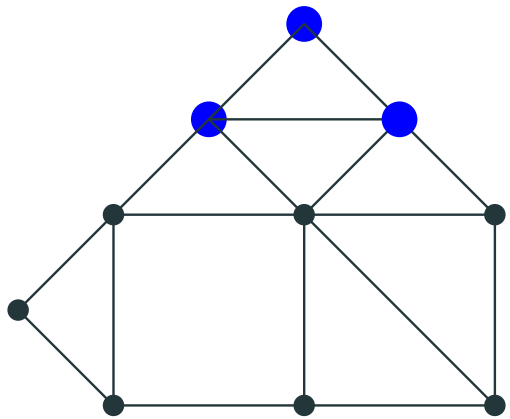
Because the structural properties of the graph do not depend on vertex names.

In particular, we will be considering various **graph invariants**
(or **graph parameters**):

mappings from the class of all graphs to some domain
(usually the set of nonnegative integers, $\mathbb{Z}_+$)

such that any two isomorphic graphs are assigned the same value.

Example: the **clique number**, defined as the maximum cardinality of a clique (set of pairwise adjacent vertices) in G



As we have already seen in some of the previous lectures, some of the most basic and yet most useful optimization problems on graphs are **domination problems.**

They occur in all contexts where we are dealing with a binary relation on a finite set V

(for example, the relation of friendship on a set of people)

and we want to find a smallest subset of a given set,

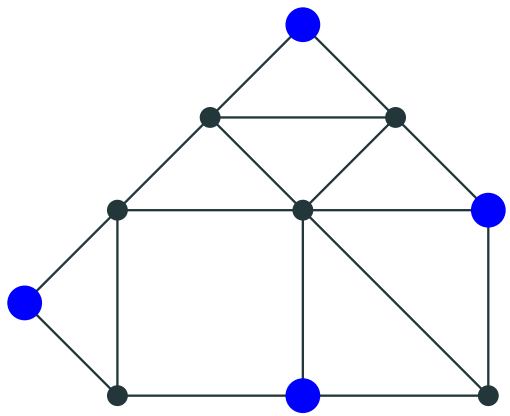
the elements of which are in relation with a prescribed proportion or part of the entire set.

Domination – the basic definition

A **dominating set** in a graph $G = (V, E)$:

a set $S \subseteq V$ such that each vertex not in S has at least one neighbor in S.

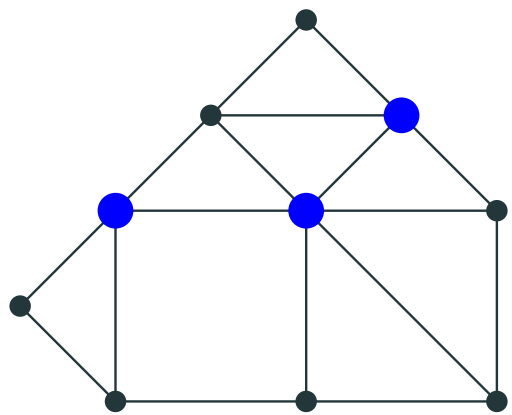
Example:



A **dominating set** in a graph $G = (V, E)$:

a set $S \subseteq V$ such that each vertex not in S has at least one neighbor in S.

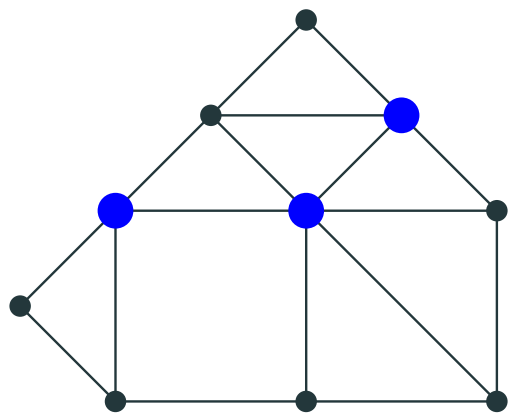
Example:



The **domination number** of a graph G is denoted by $\gamma(G)$ and defined as the minimum cardinality of a dominating set in G.

Example:

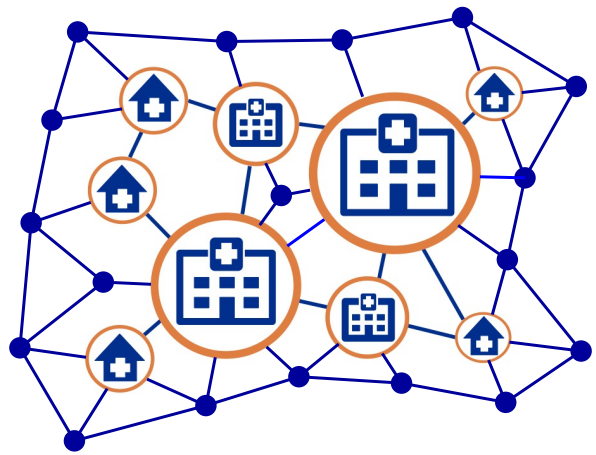
A graph G with $\gamma(G) = 3$:



Example application:

We want to set up a system of **hospital care** in the country that will be well accessible to the vast majority of the population.

- For the graph we take the **road network between the cities in the country**.
- It is reasonable to assume that the set of cities where we will build the hospitals will form a **dominating set**.



Smaller dominating set $\Longrightarrow$

lower costs for the design, construction, and maintenance of facilities.

The associated optimization problem is as follows:

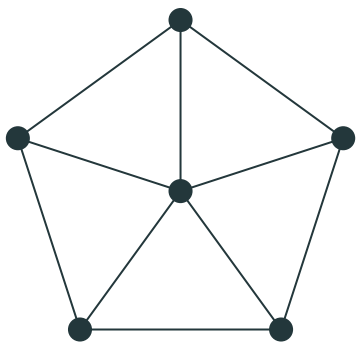
Minimum Dominating Set	
Input:	A graph G.
Task:	Find a smallest dominating set in G.

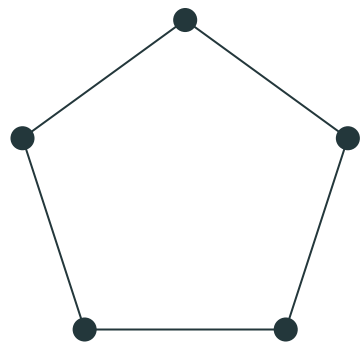
The problem also has a **weighted version**:

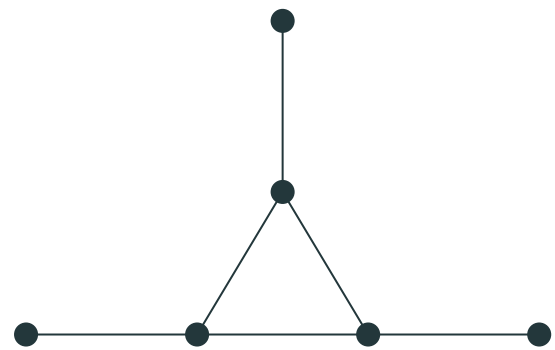
each vertex has its price; we are looking for a cheapest dominating set.

Quiz

Find the domination number of the following graphs:







How difficult is it to find an optimal solution?

The **existence** of a smallest or cheapest dominating set is guaranteed.

However:

For graphs on n vertices, already for small values of n examining all 2^n subsets of vertices is inefficient and from a practical point of view impossible.

- A graph of order 50 has $2^{50} > 10^{15}$ subsets of vertices.

  (**order** = number of vertices)
- A graph of order 100 has more than 10^{30} subsets of vertices.

A natural question arises:

Are there any fast algorithms

for Minimum Dominating Set**?**

Answer:

Most likely not.

The problem is **NP-hard**.

For the class of NP-hard problems there is a widespread belief that they are **not efficiently solvable**.

- Whether this is indeed the case is one of the most important open problems in mathematics and computer science.
- Clay Mathematics Institute is offering \$1 million for a correct solution.

Thousands of NP-hard problems are known.

NP-hardness

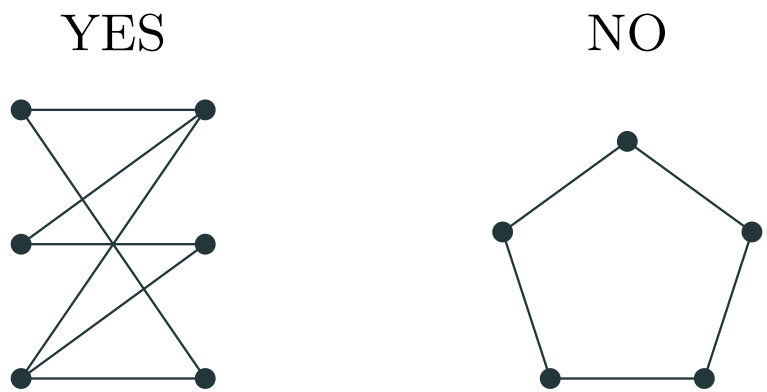
Decision problems

Decision problem: a problem for which the set of input instances divides into two sets depending on whether the answer is YES or NO.

Example:

BIPARTITENESS	
Input:	A graph $G = (V, E)$.
Question:	Is G bipartite?

A graph G is **bipartite** if its vertex set can be partitioned into two parts such that every edge has one endpoint in each of the two parts.



Any optimization problem can be converted into a decision problem.

This can be done by introducing an **additional input** that represents a lower or upper bound for the optimal solution value (a lower bound for maximization problems and an upper bound for minimization problems).

From

Minimum Dominating Set

Input: A graph G.

Task: Find a smallest dominating set in G.

we get the following decision problem:

Dominating Set

Input: A graph G, an integer k.

Question: Does G admit a dominating set of cardinality $\leq k$?

NP is the set of decision problems with the following property:

If the answer is YES, then there exists a *certificate* that enables us to verify this fact in polynomial time.

Intuitively: **NP** is the set of problems for which we can quickly verify a positive answer if we are given a solution.

Dominating Set	
Input:	A graph G, an integer k.
Question:	Does G admit a dominating set of cardinality $\leq k$?

The Dominating Set problem is in **NP**.

- If the answer is YES, then the graph G dominating set S such that $|S| \leq k$.
- Any such dominating set S can be taken as the certificate: in polynomial time, we can verify that S is a dominating set in G and that $|S| \leq k$.

NP-hard and NP-complete problems

There is a class of so-called NP-hard problems that most people *believe* are not solvable in polynomial time, but no one has ever proved that.

A problem is **NP-hard** if it is at least as hard as all problems in NP.

Intuitively:

if we could efficiently solve just one NP-hard problem, then we would be able to solve efficiently every problem whose solution we can verify quickly.

There is an important subclass of NP-hard problems – the class of NP-**complete** problems.

Definition

A problem is NP-**complete** if it is in NP and is NP-hard.

- Every NP-complete problem is a decision problem.
- These are the hardest problems in NP.

  If there exists a polynomial-time algorithm for just one NP-complete problem, then all NP-complete problems are polynomially solvable.
- Thousands of NP-complete problems are known. A polynomial algorithm for any of them is unlikely.

How do we show that a problem is NP-complete?

A, B decision problems

Definition

A **polynomial reduction** of problem A to problem B is a polynomial-time algorithm that for every instance I of A

construct an instance $J = J(I)$ for problem B such that

the answer to A given I is the same as the answer to B given J.

If there exists a polynomial reduction from problem A to problem B, we denote this by $A \to B$.

Meaning: *"Problem A is not harder than problem B."*

To show that a problem A is NP-hard,
we reduce a known NP-hard problem to A.

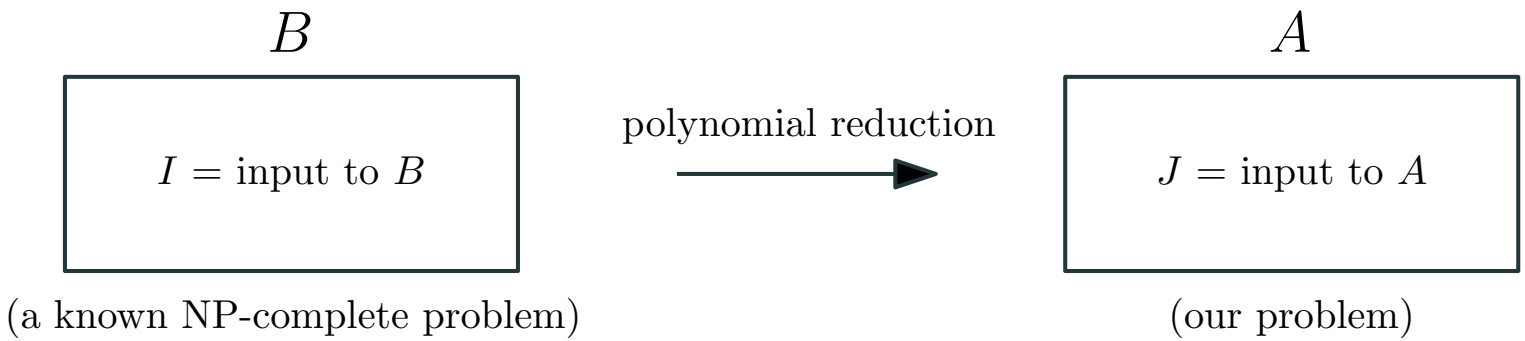
Proposition

Suppose that for a problem $A \in \mathrm{NP}$ *there exists an* NP*-complete problem* B *such that* $B \to A$.

Then A *is* NP*-complete.*

If we want to prove that a problem A is NP-**complete**, we do two things:

1. Verify that A is in NP (this is usually the easy part – we just need to *describe* a suitable certificate), and
2. Find a problem B that is known to be NP-complete and **reduce (= find a polynomial reduction from) B to A.**



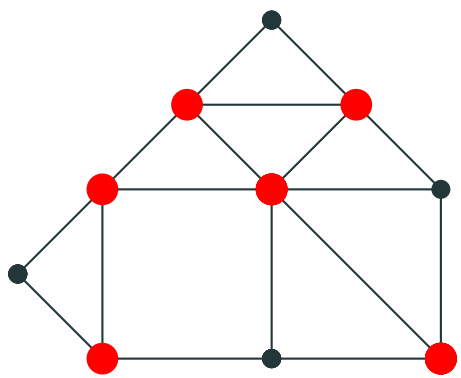
B could be any of the known NP-complete problems, for example
3-SAT, Independent Set, Vertex Cover, Clique, Colorability, etc.

NP-completeness of Dominating Set

We will reduce from the following problem.

Vertex Cover	
Input:	A graph $G = (V, E)$, integer k.
Question:	Does G admit a vertex cover of cardinality $\leq k$?

vertex cover: a set $C \subseteq V$ such that every edge $e \in E$ has an endpoint in C.



Proposition (Karp, 1972)

Vertex Cover *is* NP*-complete.*

Quiz

Prove or disprove:

Every vertex cover in a graph is also a dominating set.

Proposition

Dominating Set *is* NP*-complete.*

Proof:

1. Dominating Set $\in$ NP.

2. Vertex Cover $\rightarrow$ Dominating Set

$I = (G = (V, E), k)$ instance for Vertex Cover $\mapsto$

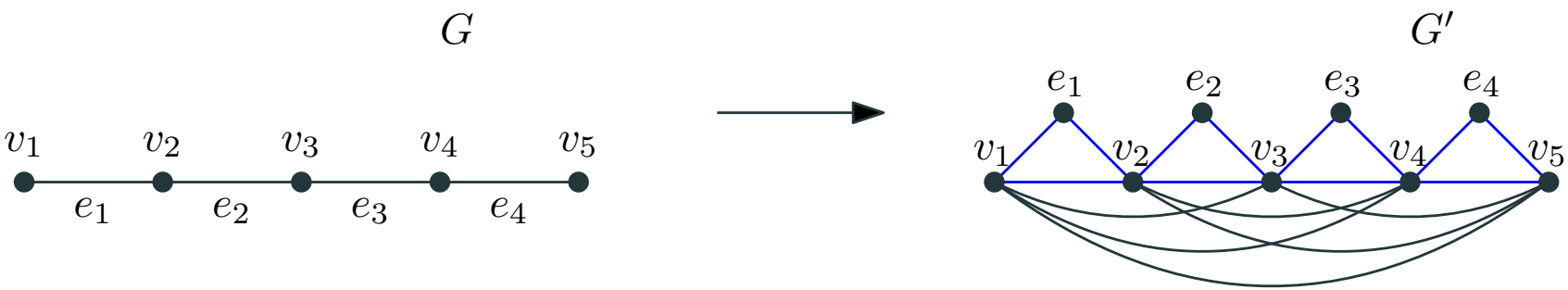
$J(I) = (G' = (V', E'), k)$ instance for Dominating Set where:

- $V' = V \cup E$,
- V is a clique in G',
- E is an independent set in G',
- $ve \in E(G') \Leftrightarrow v \in V, e \in E, v$ is an endpoint of e.

It can be shown:

G has a vertex cover of size $\leq k \iff$

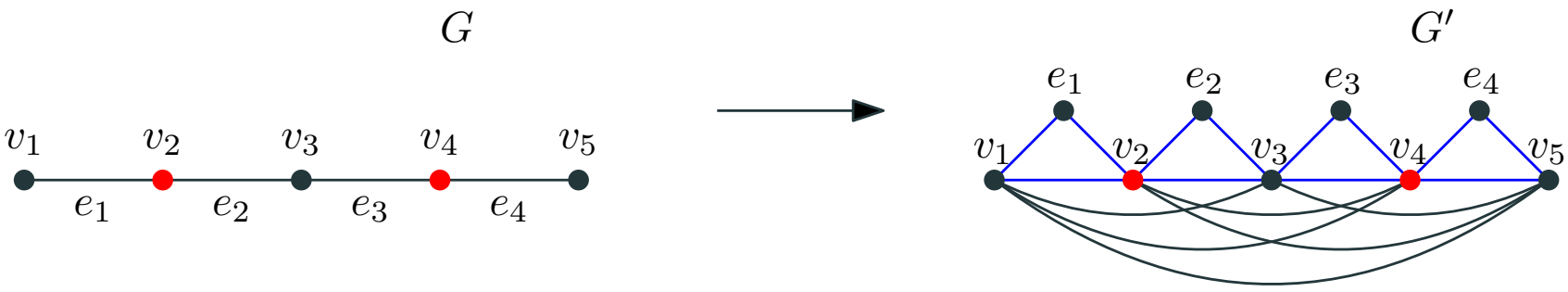
G' has a dominating set of size $\leq k$.



It can be shown:

G has a vertex cover of size $\leq k \iff$

G' has a dominating set of size $\leq k$.



Corollary

Minimum Dominating Set *is* NP*-hard.*

Solving the Minimum Dominating Set problem to optimality is, therefore, hopeless ...

But what if we are satisfied with a solution that is at most a factor of k worse than an optimal one?

- Here k is some constant.
  For example:
  $k = 1.1, \quad k = 2, \quad k = 10$

Unfortunately, it is known that **even in this case efficient algorithms most likely do not exist.**

The big picture

For some practically and theoretically relevant graph problems,
efficient algorithms are known.

This is the case for:

- shortest path problems
- minimum spanning trees
- maximum matchings

Many other problems behave similarly as domination problems:
no efficient algorithms are known.

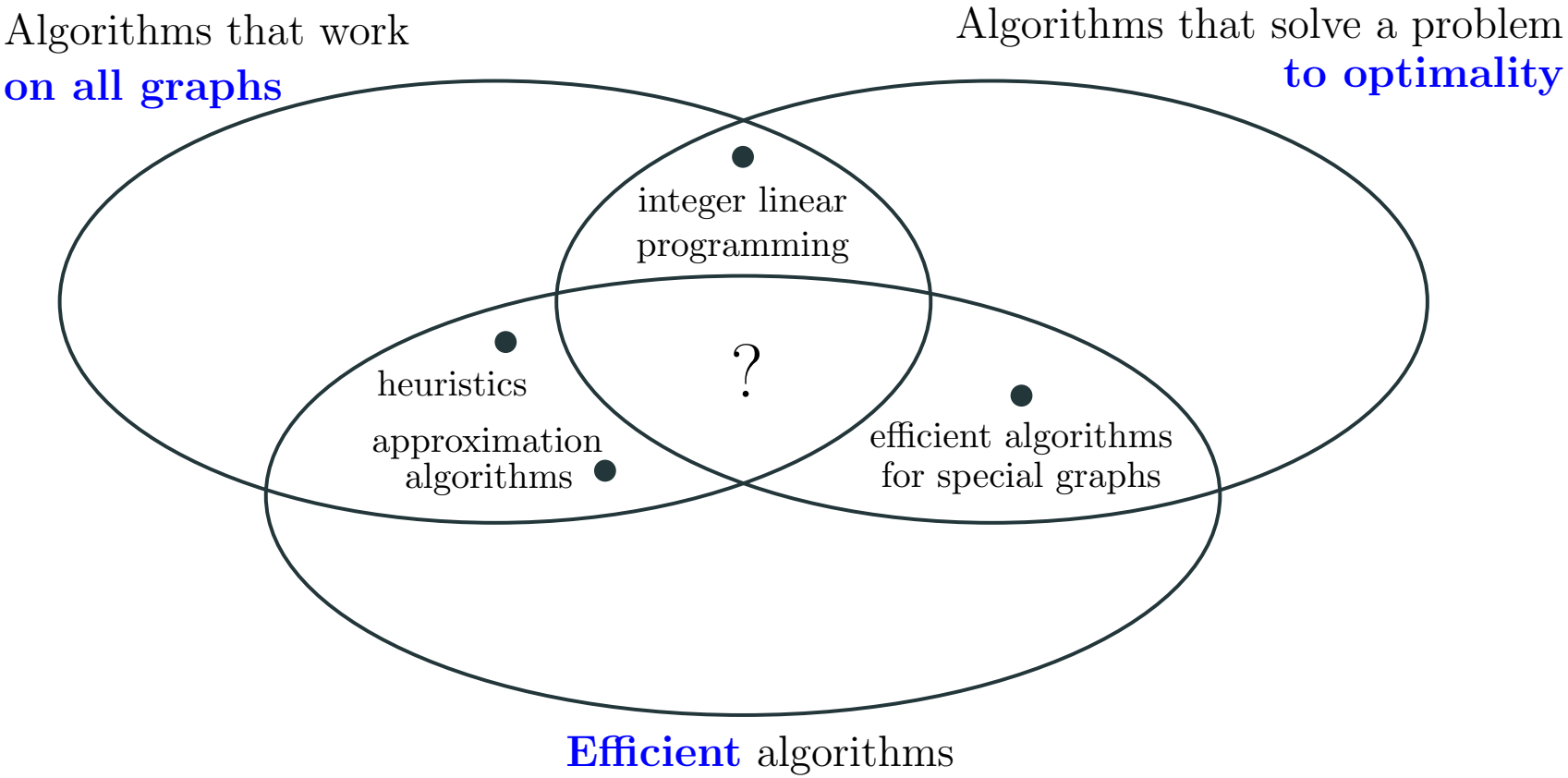
This is the case for:

- the vertex cover problem
- the graph coloring problem
- the traveling salesman problem

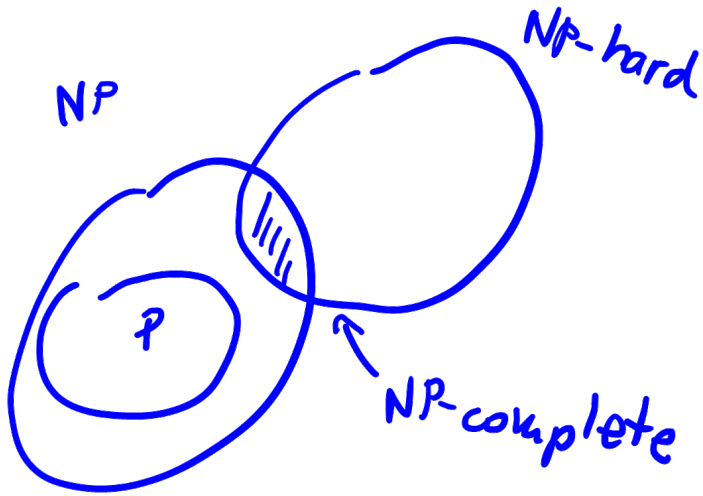
The theory, then, says that many graph problems are computationally very difficult.

In practice, however, problems must be solved!

Various approaches are in use:



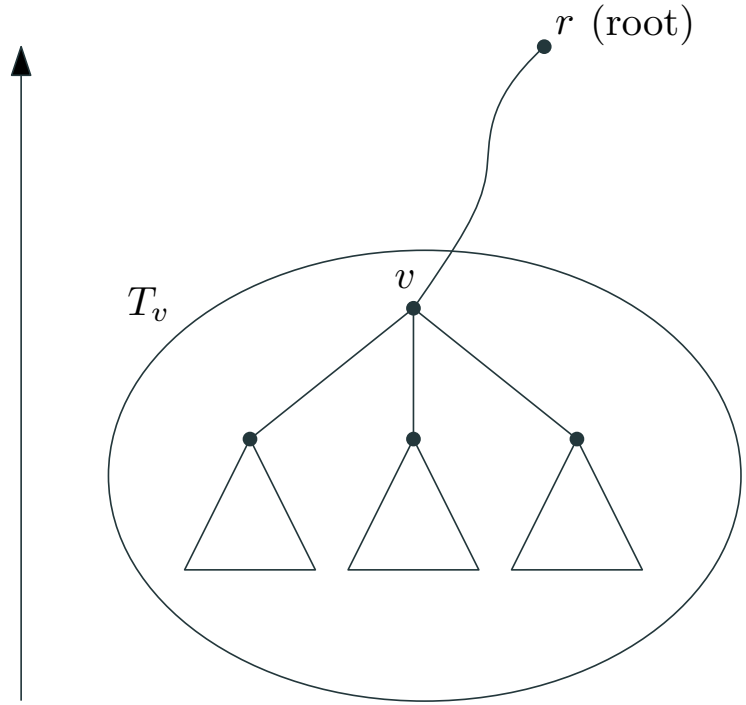
Efficient algorithms for special cases



Dynamic programming on trees

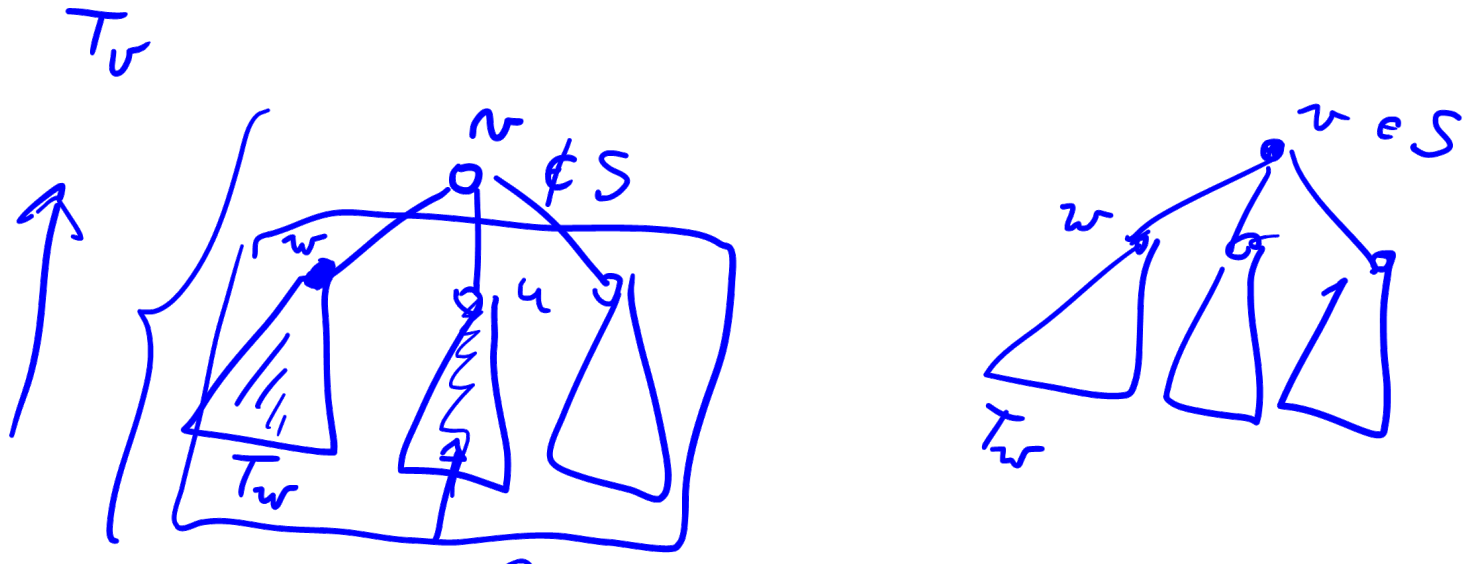
The Minimum Dominating Set problem can be solved optimally on trees, by a linear-time **dynamic programming** algorithm:

1. Root the tree T at an arbitrary vertex r.



2. Traverse the tree from the leaves to the root and recursively compute, for each vertex $v \in V(T)$, the following quantities:

$$\gamma_v = \min\{\gamma_v^+, \gamma_v^-\}$$

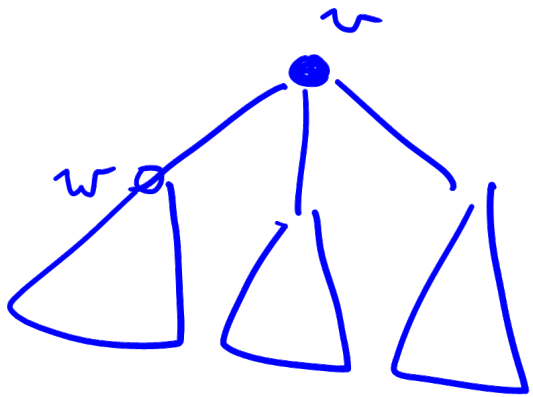


$$\gamma_v^- = \min_w \left(\gamma_w^+ + \sum_{u \neq w} \gamma(T_u) \right)$$

$$\gamma_v^+ = 1 + \sum_w \gamma_w^0$$

$$\gamma_w^0 =$$

2. Traverse the tree from the leaves to the root and recursively compute, for each vertex $v \in V(T)$, the following quantities:
   - γ_v, the minimum cardinality of a dominating set S in T_v
   - γ_v^+, the minimum cardinality of a dominating set S in T_v such that $v \in S$
   - γ_v^-, the minimum cardinality of a dominating set S in T_v such that $v \notin S$
   - γ_v^0, the minimum cardinality of a set $S \subseteq V(T_v)$ that dominates all vertices of T_v except possibly v



$\gamma_v^0 \leq ?\ \gamma_v^-$

$$\gamma_v^0 = \min\left\{ \sum_{\substack{w \\ v \notin S}} \gamma_w ,\ 1 + \sum_{\substack{w \\ v \in S}} \gamma_w^0 \right\}$$

Generalizations

The same idea works for many optimization problems on trees.

This idea has been generalized in multiple ways in the literature:

- first, by generalizing the **structure**, by considering graphs that **"resemble trees"** in the sense that
  they can be recursively decomposed along small separators, or
  they admit recursive constructions that can be described by trees

  (like the class of **cographs** *discussed by Christophe Crespelle on Wednesday)*

- second, by generalizing the **family of problems** that can be efficiently solved given a recursive construction of the graph.

  For each structure there is a particular **logic that can be used for describing problems**.

Such statements are known as **algorithmic metatheorems.**

In particular, for Minimum Dominating Set and related problems the following approaches apply:

1. bounded **treewidth**: Courcelle's theorem (early 1990s); Bonomo-Braberman, Gonzalez (2022)

2. bounded **clique-width**: Courcelle, Makowsky, Rotics (2000); Baghirova, Gonzalez, Ries, Schindl (2022)

3. bounded **mim-width**: Bergougnoux, Dreier, Jaffke (2023)

A good starting point to learn about (classical) graph width parameters is the survey paper **Width Parameters Beyond Tree-width and their Applications** by Hliněný, Oum, Seese, and Gottlob, *The Computer Journal*, 2008.

clique-width of a graph $G = (V, E)$ = smallest number of labels in a k-expression constructing (a graph isomorphic to) G

A k-**expression** is an algebraic expression building a graph together with labels $\ell(v) \in \{1, \ldots, k\}$ for all $v \in V$,

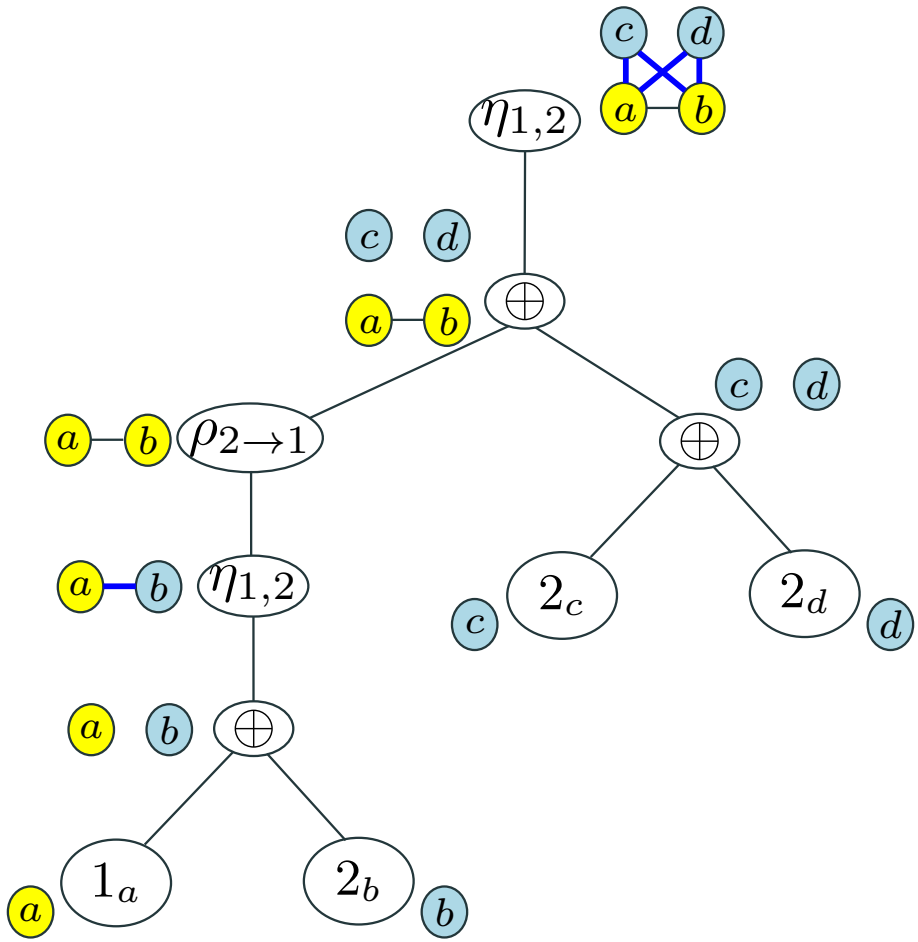
using the following operations:

- ℓ_v, $\ell \in \{1, \ldots, k\}$: creating a new one-vertex graph with vertex v labeled ℓ,
- $G \oplus H$: disjoint union
- $\eta_{i,j}(G)$ for $i, j \in \{1, \ldots, k\}, i \neq j$: add edges
- $\rho_{i \to j}(G)$ for $i, j \in \{1, \ldots, k\}, i \neq j$: relabel

Example:

The following expression builds a complete graph minus an edge using only labels 1 and 2 (yellow and blue, respectively):

$$\eta_{1,2}\left(\rho_{2\to 1}\left(\eta_{1,2}\left(1(a)\oplus 2(b)\right)\right)\oplus\left(2(c)\oplus 2(d)\right)\right)$$



Example:

The clique-width of any P_4-free graph (cograph) is at most 2.

Proof:

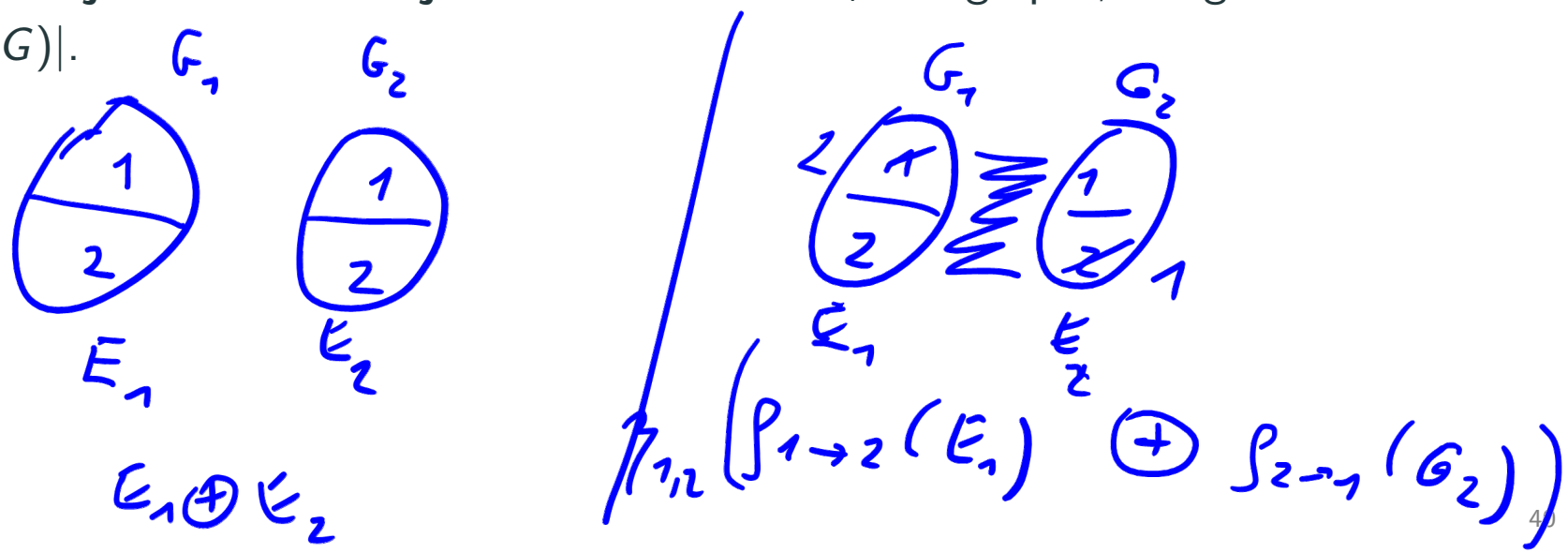
It follows from the fact that every P_4-free graph with at least two vertices is either a **disjoint union** or a **join** of two smaller P_4-free graphs, using induction on $|V(G)|$.

Example:

The clique-width of any P_4-free graph (cograph) is at most 2.

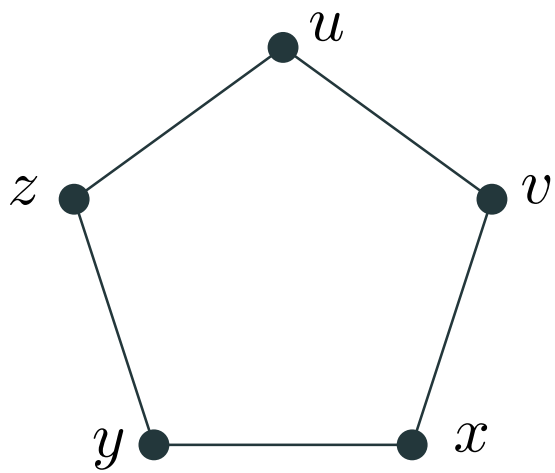
Proof:

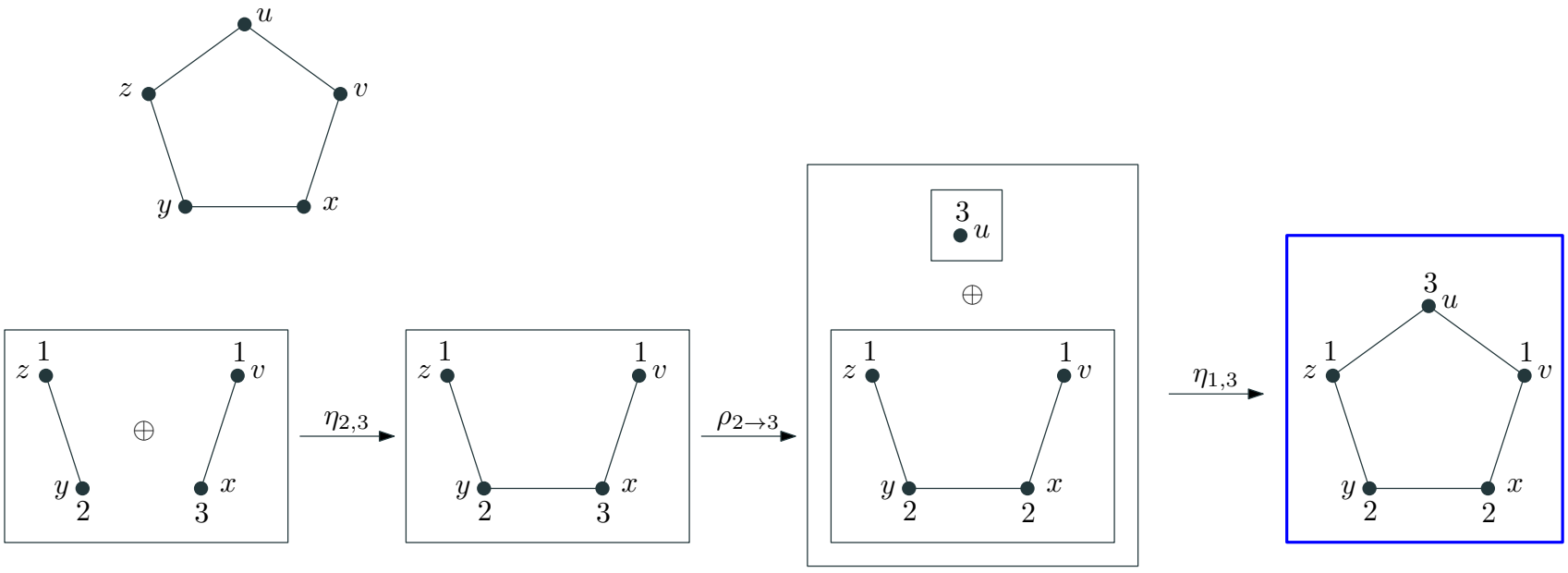
It follows from the fact that every P_4-free graph with at least two vertices is either a **disjoint union** or a **join** of two smaller P_4-free graphs, using induction on $|V(G)|$.



Example:

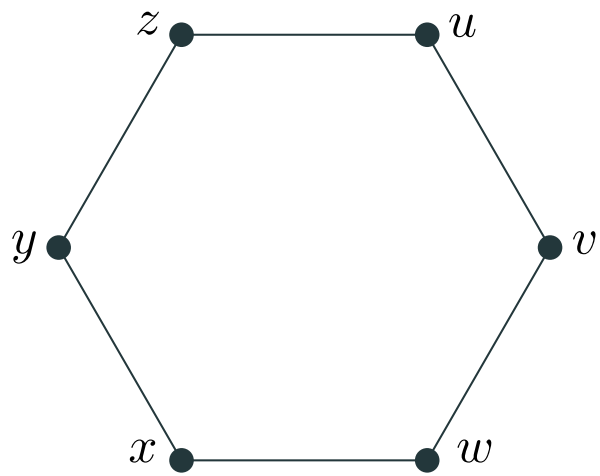
The clique-width of the following graph is at most 3:



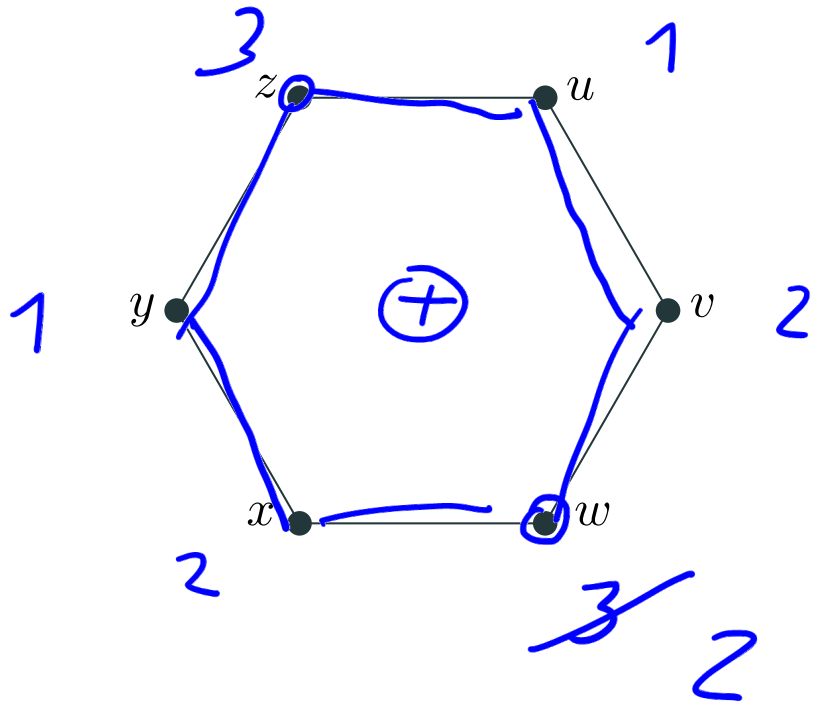

u
z
v
y
x
1
z
1
v
y
2
x
3
⊕
$\eta_{2,3}$
1
z
1
v
y
2
x
3
$\rho_{2\to 3}$
3
u
⊕
1
z
1
v
y
2
x
2
$\eta_{1,3}$
3
u
1
z
1
v
y
2
x
2

Quiz

Show that the clique-width of the following graph is at most 3.



Show that the clique-width of the following graph is at most 3.



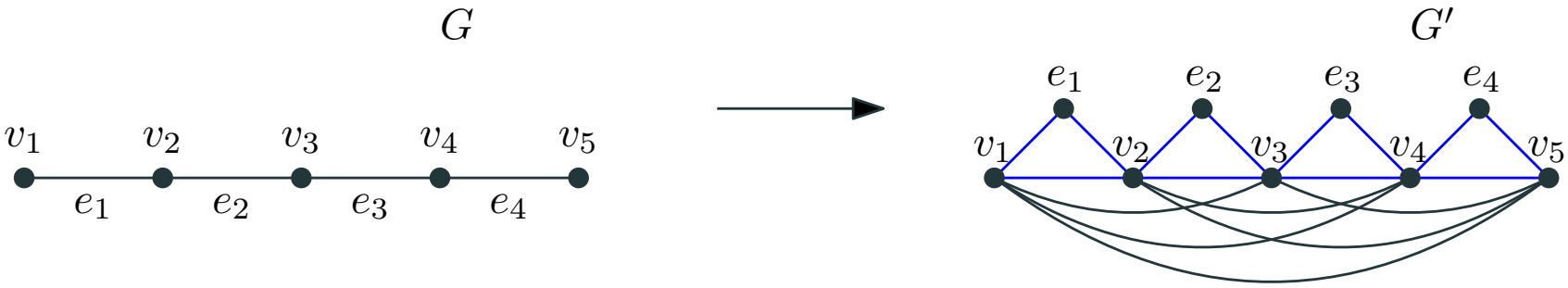
Many algorithmic decision or optimization problems on graphs that are NP-hard for general graphs can be solved **in polynomial time on graph classes of bounded clique-width**

(or even in linear time if a k-expression is known).

Courcelle, Makowsky, and Rotics (2000) (and others)

Using clique-width: Example 1

Recall the graphs obtained by the proof of NP-hardness of Minimum Dominating Set:

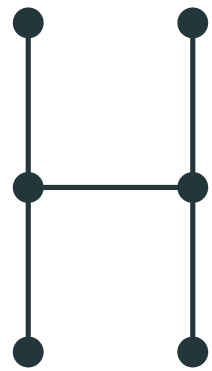


The graphs are **split**: they admit a partition of the vertex set into a **clique** and an **independent set**

Thus, Minimum Dominating Set is NP-hard even when restricted to the class of split graphs.

The clique-width of split graphs is known to be unbounded.

It remains unbounded even for the class of split graphs that are H-**free**: they do not contain H **as an induced subgraph**.



H

This was shown by Brandstädt, Dabrowski, Huang, Paulusma in 2016.

Nevertheless, Minimum Dominating Set can be solved in polynomial time in this class of graphs.

Let G be a split graph with a clique C and independent set I.

We would like to solve the Minimum Dominating Set on G.

Three observations:

1. We may assume that G has no isolated vertices.
2. There exists a minimum dominating set S such that $S \subseteq C$.
3. If there exist two distinct vertices $x, y \in C$ such that $N[x] \subseteq N[y]$, then $\gamma(G) = \gamma(G - x)$ and any dominating set of $G - x$ contained in $C \setminus \{x\}$ is a dominating set in G.
   Thus, vertex x can be safely deleted.

So we may assume that

no two vertices in C have comparable closed neighborhoods.

In this case, it can be shown that the clique-width becomes bounded by 5.

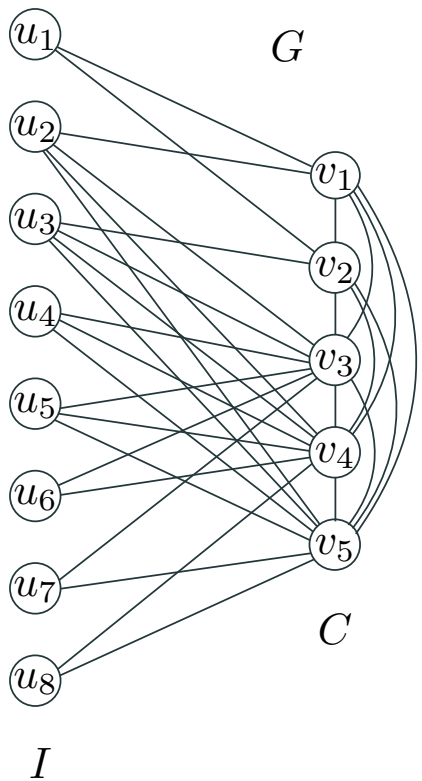
Reference: Boros, Gurvich, M. Characterizing and decomposing classes of threshold, split, and bipartite graphs via 1-Sperner hypergraphs, *J. Graph Theory* 94 (2020) 364–397.

Proof idea

We use a decomposition result for the hypergraphs that describe the edges between the clique and the independent set.

This allows to recursively decompose the graph.

Given a split graph G with a split partition (C, I), define a hypergraph $\mathcal{H} = (V, E)$ with $V = I$ and $E = \{N(v) \cap I : v \in C\}$.



$$A^{\mathcal{H}} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccccccc} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 & u_8 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Since the neighborhoods in I of vertices in C are pairwise incomparable, $\mathcal{H}$ is **Sperner**, that is, no hyperedge contains another one.

Sperner hypergraphs were studied under many different names:

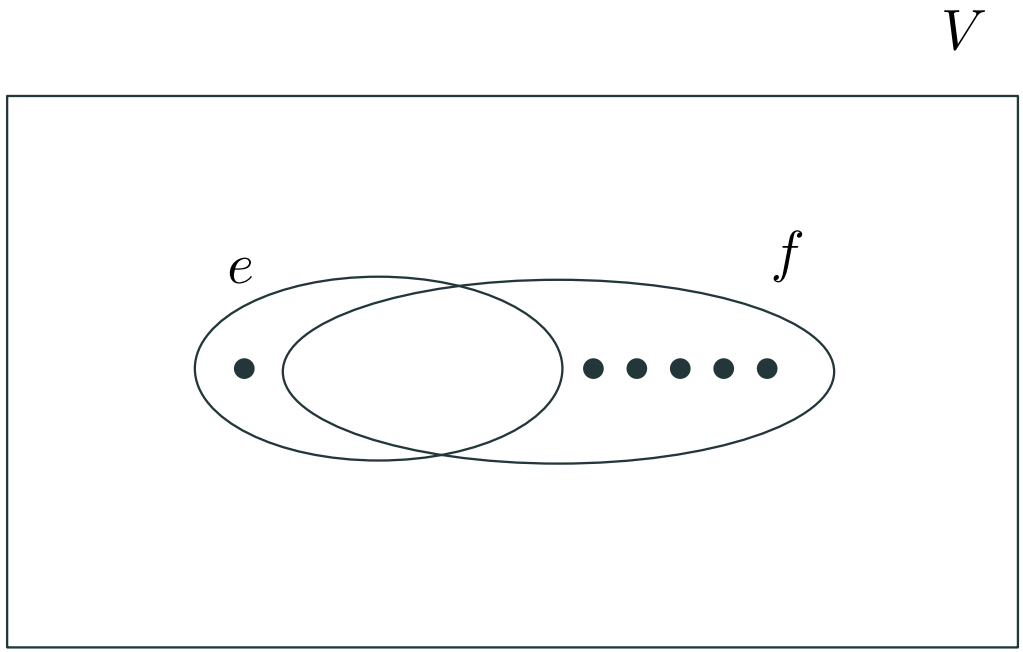
- **simple hypergraphs** (by Berge),
- **clutters** (by Billera and by Edmonds and Fulkerson)
- **coalitions** (in the game theory literature)

Since G is H-free, $\mathcal{H}$ is also 1-**Sperner**.

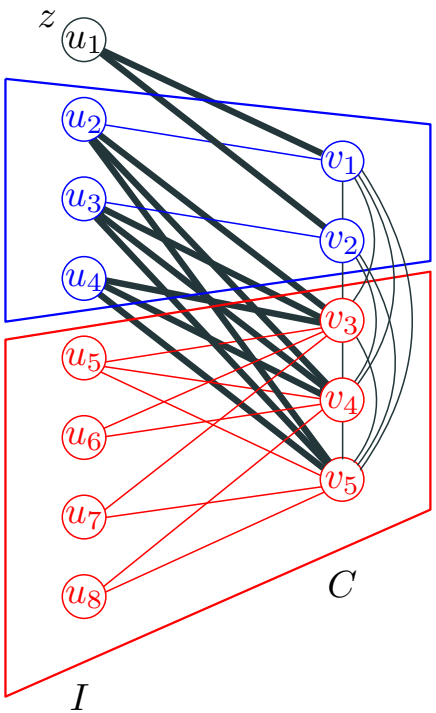
A hypergraph $\mathcal{H}$ is 1-**Sperner** if every two distinct hyperedges e and f satisfy

$$\min\{|e \setminus f|, |f \setminus e|\} = 1\,,$$

that is, if **for every two hyperedges** the smallest of the two set differences is of size one.



There is a composition theorem for 1-Sperner hypergraphs.

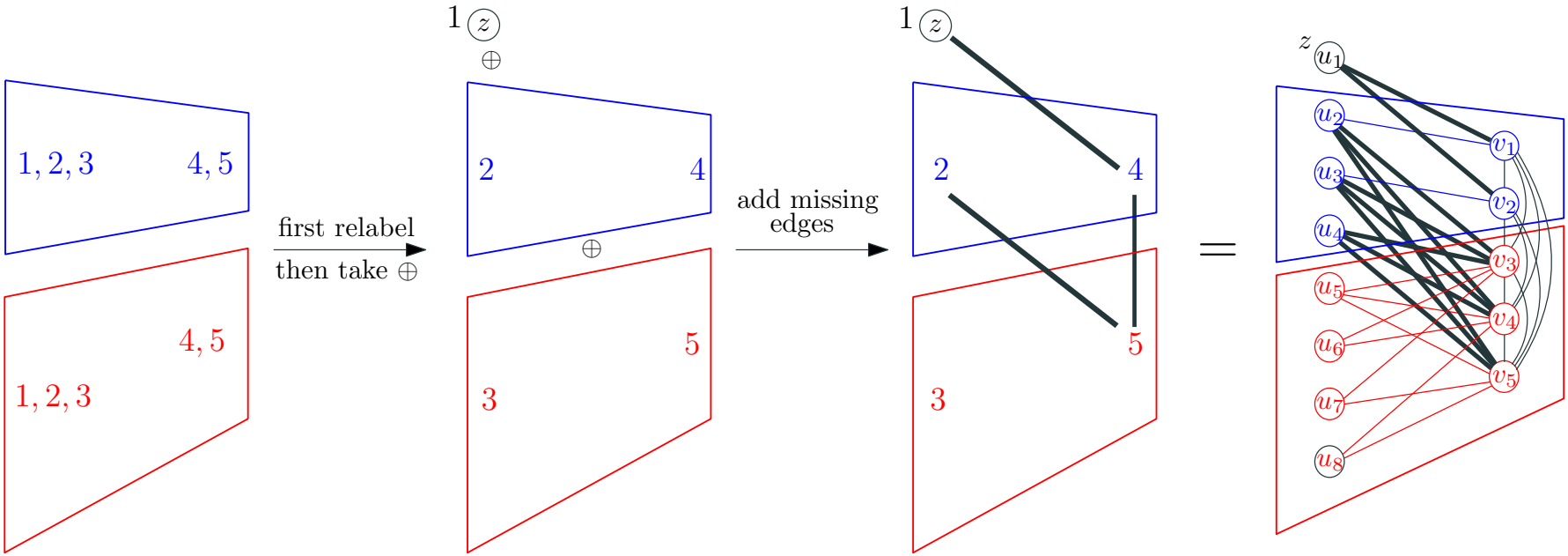


$$A^{\mathcal{H}} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \overset{z}{\overset{\shortparallel}{u_1}} \quad u_2 \quad u_3 \quad u_4 \quad u_5 \quad u_6 \quad u_7 \quad u_8 \\ \left(\begin{array}{c|ccc|cccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{array}\right) \end{array}$$

incidence matrix of hypergraph $\mathcal{H} = \mathcal{H}_1 \odot \mathcal{H}_2$

$$A^{\mathcal{H}_1} = \begin{array}{c} \\ v_1 \\ v_2 \end{array} \begin{array}{c} u_2 \quad u_3 \quad u_4 \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$A^{\mathcal{H}_2} = \begin{array}{c} \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} u_5 \quad u_6 \quad u_7 \quad u_8 \\ \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$


1, 2, 3
4, 5
4, 5
1, 2, 3
first relabel
then take ⊕
1
z
⊕
2
4
⊕
5
3
add missing
edges
1
z
2
4
5
3
=
z
u_1
u_2
u_3
u_4
u_5
u_6
u_7
u_8
v_1
v_2
v_3
v_4
v_5

This result leads to **efficient algorithms** for three basic variants of the **dominating set problem** in the class of H-free split graphs.

Given a graph $G = (V, E)$, a set $S \subseteq V$ is:

- a **dominating set** if every vertex in $V \setminus S$ has a neighbor in S,
- a **total dominating set** if every vertex has a neighbor in S,
- a **connected dominating set** if it is a dominating set that induces a connected subgraph of G.

Theorem

The problems of finding a minimum dominating set / total dominating set / connected dominating set are solvable in time $\mathcal{O}(|V(G)|^3)$ in the class of H-free split graphs.

This result is sharp in the sense that all three problems are known to be NP-hard:

- in the class of split graphs,
- in the class of H-free graphs.

Using clique-width: Example 2

For the end, let us consider the following variant of domination:

A vertex $v \in V$ dominates itself and all its neighbors

Given a graph $G = (V, E)$, a set $S \subseteq V$ is an **efficient dominating set** in G if every vertex in G is dominated by **exactly one** vertex in S:

$$|N[v] \cap S| = 1$$

for all $v \in V$.

- introduced by Biggs 1973 (under the name "perfect code")

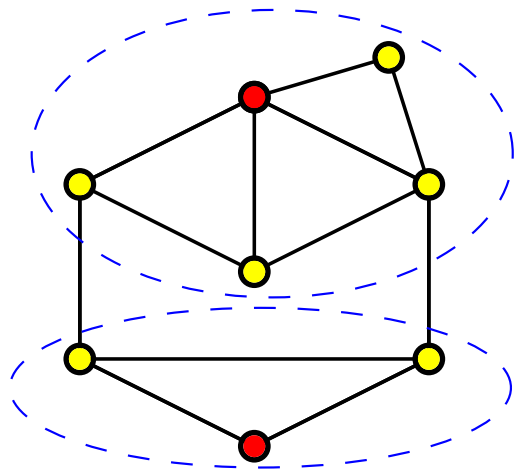
Equivalently:

- S is an independent set of vertices such that
- every vertex outside S has a unique neighbor in S.

Equivalently:

$$\{N[v] \mid v \in S\}$$

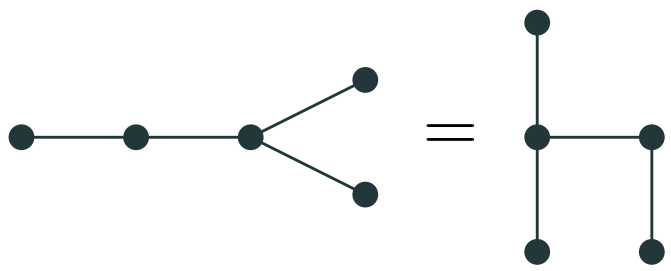
forms a partition of V.



However, not every graph has an efficient dominating set.

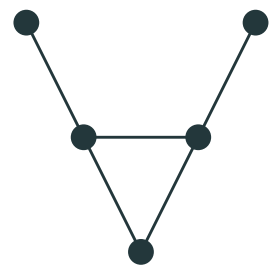
Quiz

Show that neither of the following two graphs has an efficient dominating set:



fork

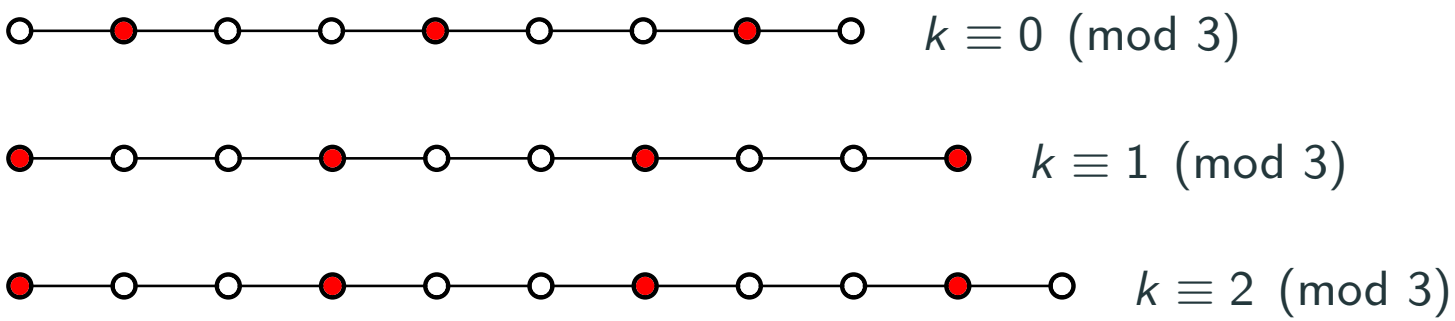
chair



bull

All paths contain efficient dominating sets:

P_k



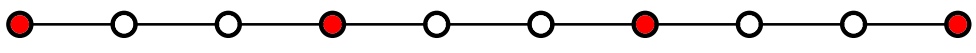
Cycle C_k contains an efficient dominating set

$\iff k \equiv 0 \pmod 3$.

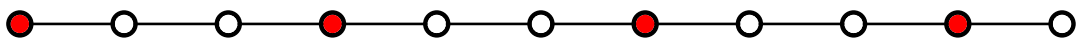
All paths contain efficient dominating sets:

P_k

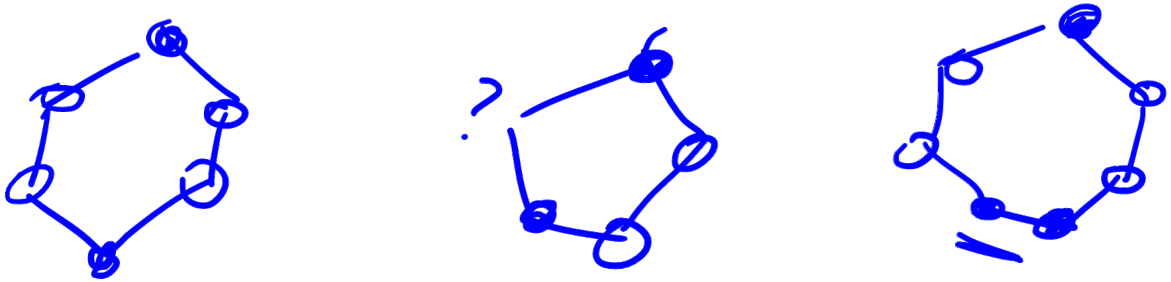
$k \equiv 0 \pmod 3$



$k \equiv 1 \pmod 3$



$k \equiv 2 \pmod 3$

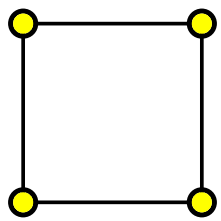


G is **efficiently dominatable (ED)** if it contains an efficient dominating set.

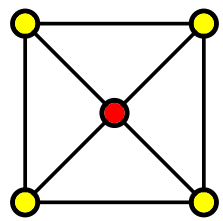
Determining whether G is efficiently dominatable is **NP-complete**, even for:

- planar cubic graphs,
- planar bipartite graphs,
- chordal graphs,
- chordal bipartite graphs,
- line graphs of planar bipartite graphs of max degree three.

The class of efficiently dominatable graphs is not closed under vertex deletion:



not ED



ED

What are the graphs in which every induced subgraph has an efficient dominating set?

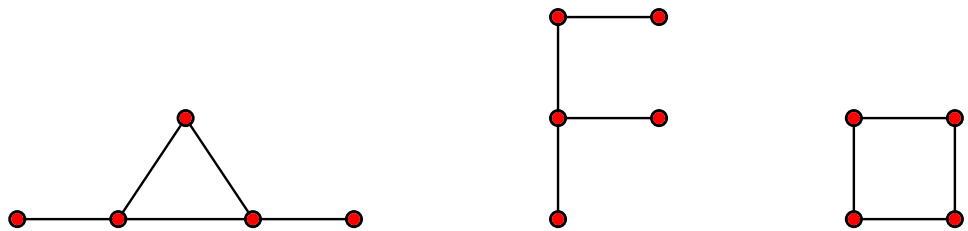
G is hereditary efficiently dominatable (HED) if every induced subgraph of G is ED.

It turns out that these are precisely the graphs not containing the bull, the fork, or a cycle C_k with $k \not\equiv 0 \bmod 3$, as an induced subgraph.

Reference: M. Hereditary efficiently dominatable graphs, *J. Graph Theory* 73 (2013) 400–424.

Proposition

Every HED graph is (bull, fork, C_{3k+1}, C_{3k+2})-free.



The converse holds as well.

To prove this, we study the structure of (bull, fork, C_4)-free graphs.

Theorem

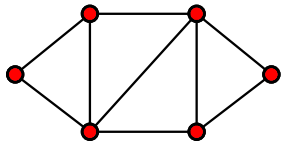
Let G be a (bull, fork, C_4)-free graph. Then, G can be built from

paths and cycles

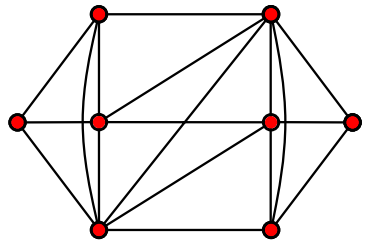
by applying a sequence of the following operations:

- *disjoint union of two graphs,*
- *adding a true twin,*
- *adding a universal vertex,*
- *raft expansion,*
- *semi-raft expansion.*

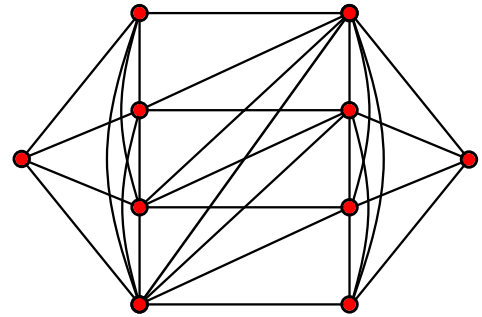
Rafts of order 2, 3, and 4:



R_2

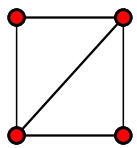


R_3

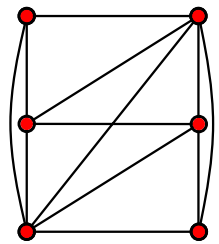


R_4

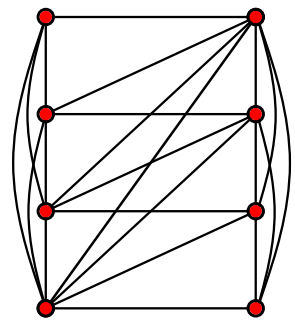
Semi-rafts of order 2, 3, and 4:



S_2

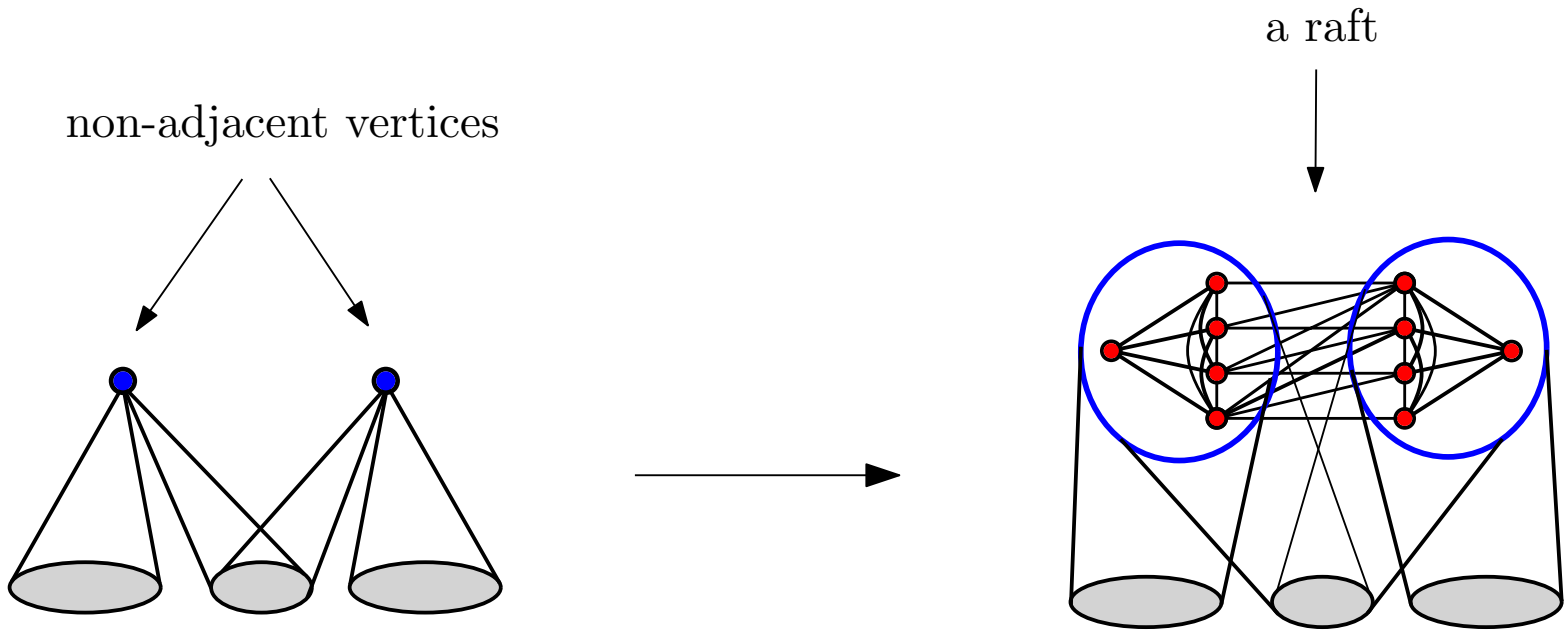


S_3

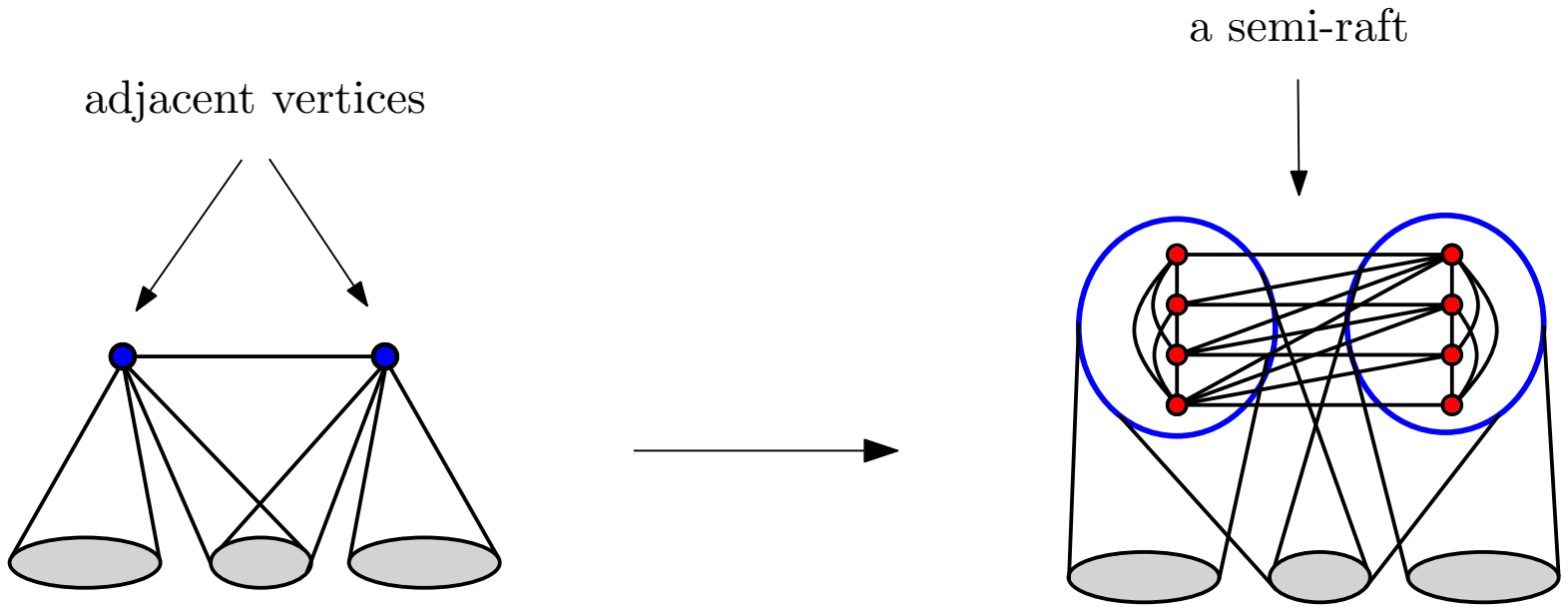


S_4

Raft expansion:



Semi-raft expansion:



Corollary

The class of HED graphs equals the class of (bull, fork, C_{3k+1}, C_{3k+2})-free graphs.

HED graphs can be can be recognized **in linear time**:

1. These graphs have bounded clique-width and a k-expression can be computed in linear time (this follows from a more general result by Brandstädt, Hoàng, Le, 2003).

2. For every k, every (decision or optimization) problem expressed in MSOL with quantifiers over vertex sets is solvable in linear time, given a k-expression of the input graph (Courcelle, Makowsky, Rotics, 2000).

3. We can express the defining property in MSOL (Monadic Second Order Logic) with quantifiers over vertices and vertex subsets, as follows:

$$(\forall X \subseteq V)(\exists S \subseteq X)(\mathit{EfficientDominating}(S, X)),$$

where

$$\mathit{EfficientDominating}(S, X) := (\forall x \in X)((\exists y \in S)(y = x \vee E(y, x)) \wedge$$

$$(\forall y \in S)(\forall z \in S)((y = x \vee E(y, x)) \wedge (z = x \vee E(z, x)) \rightarrow y = z)).$$

The same approach can also be used to compute an **efficient dominating set of maximum or minimum weight** in a given vertex-weighted HED graph.

Thank you!

Questions?