Decomposing Subcubic Graphs into Claws, Paths or Triangles

Laurent Bulteau Guillaume Fertin **Anthony Labarre**
Romeo Rizzi Irena Rusu

Séminaire QuaResMi

October 14th, 2021

Université
Gustave Eiffel

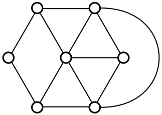
The graph decomposition problem

Given a set S of graphs, an S-**decomposition** of a graph $G = (V, E)$ is a partition of E into subgraphs, all of which are isomorphic to a graph in S.

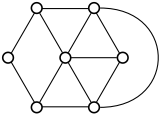
The graph decomposition problem

Given a set S of graphs, an S-**decomposition** of a graph $G = (V, E)$ is a partition of E into subgraphs, all of which are isomorphic to a graph in S.

Example



$S = \{$, $\}$

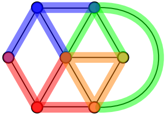


$S =$ connected graphs on 4 edges

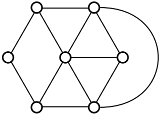
The graph decomposition problem

Given a set S of graphs, an S-**decomposition** of a graph $G = (V, E)$ is a partition of E into subgraphs, all of which are isomorphic to a graph in S.

Example



$S = \{$⅄, △$\}$

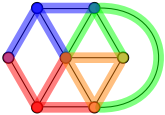


$S =$ connected graphs on 4 edges

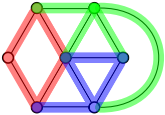
The graph decomposition problem

Given a set S of graphs, an S-**decomposition** of a graph $G = (V, E)$ is a partition of E into subgraphs, all of which are isomorphic to a graph in S.

Example



$S = \{$⅄, △$\}$

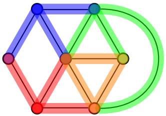


$S =$ connected graphs on 4 edges

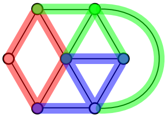
The graph decomposition problem

Given a set S of graphs, an S-**decomposition** of a graph $G = (V, E)$ is a partition of E into subgraphs, all of which are isomorphic to a graph in S.

Example



$S = \{$⅄, △$\}$



$S =$ connected graphs on 4 edges

S-DECOMPOSITION
Input: a graph $G = (V, E)$, a set S of graphs.
Question: does G admit an S-decomposition?

Motivations

Edge-partition problems appear in surprisingly diverse areas:

- database anonymisation [1];
- traffic grooming [7];
- graph drawing [4];
- . . .

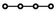
What is known?

- Old problem (earliest reference we found is from 1847);

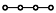
What is known?

- Old problem (earliest reference we found is from 1847);
- Our starting point is the following piece of bad news: S-DECOMPOSITION is NP-complete, even when S contains a single connected graph with at least three edges [2].

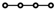
What is known?

- Old problem (earliest reference we found is from 1847);
- Our starting point is the following piece of bad news: S-DECOMPOSITION is NP-complete, even when S contains a single connected graph with at least three edges [2].
- Are there easy cases when we restrict ourselves to connected subgraphs with three edges? (i.e. $S = \{$, , $\}$)

What is known?

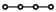
- Old problem (earliest reference we found is from 1847);
- Our starting point is the following piece of bad news: S-DECOMPOSITION is NP-complete, even when S contains a single connected graph with at least three edges [2].
- Are there easy cases when we restrict ourselves to connected subgraphs with three edges? (i.e. $S = \{$, , $\}$)
- It turns out that the answer is yes if the input graph is subcubic (all degrees ≤ 3);

Our contributions

Here is a summary of what is known about decomposing graphs using subsets of { , , }:

Allowed subgraphs			Complexity according to graph class		
			strictly subcubic	cubic	arbitrary
✓					NP-complete [3, Theorem 3.5]
	✓			$O(1)$ (impossible)	NP-complete [5]
		✓		in P [6]	NP-complete [3, Theorem 3.4]
✓	✓				NP-complete [3, Theorem 3.5]
✓		✓			NP-complete [3, Theorem 3.1]
	✓	✓			NP-complete [3, Theorem 3.4]
✓	✓	✓			NP-complete [3, Theorem 3.1]

Our contributions

Here is a summary of what is known about decomposing graphs using subsets of { , , }:

Allowed subgraphs			Complexity according to graph class		
			strictly subcubic	cubic	arbitrary
✓			in P	in P	NP-complete [3, Theorem 3.5]
	✓		in P	$O(1)$ (impossible)	NP-complete [5]
		✓	NP-complete	in P [6]	NP-complete [3, Theorem 3.4]
✓	✓		in P	in P	NP-complete [3, Theorem 3.5]
✓		✓	NP-complete	NP-complete	NP-complete [3, Theorem 3.1]
	✓	✓	NP-complete	in P	NP-complete [3, Theorem 3.4]
✓	✓	✓	NP-complete	NP-complete	NP-complete [3, Theorem 3.1]

our contributions

Decomposing strictly subcubic graphs

- G is **subcubic** if all vertices have degree ≤ 3;

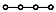
Decomposing strictly subcubic graphs

- G is **subcubic** if all vertices have degree ≤ 3;
- G is **strictly subcubic** if it is subcubic and it has a vertex of degree 1 or 2. In this case, we show that:

Decomposing strictly subcubic graphs

- G is **subcubic** if all vertices have degree ≤ 3;
- G is **strictly subcubic** if it is subcubic and it has a vertex of degree 1 or 2. In this case, we show that:
    - decomposing G using or or both is in P;

Decomposing strictly subcubic graphs

- G is **subcubic** if all vertices have degree ≤ 3;
- G is **strictly subcubic** if it is subcubic and it has a vertex of degree 1 or 2. In this case, we show that:
  - decomposing G using or or both is in P;
  - otherwise (i.e. as soon as we allow 's) it is NP-complete.

Decomposing strictly subcubic graphs

- *G* is **subcubic** if all vertices have degree ≤ 3;
- *G* is **strictly subcubic** if it is subcubic and it has a vertex of degree 1 or 2. In this case, we show that:
  - decomposing *G* using ⅄ or △ or both is in P;
  - otherwise (i.e. as soon as we allow ∘–∘–∘–∘'s) it is NP-complete.
- The △ case is trivial: *G* admits a △ -decomposition if and only if it is a disjoint union of △'s.

⅄-DECOMPOSITION, strictly subcubic

- Simple algorithm: extract a ⅄ at each step as long as possible;

⅄-DECOMPOSITION, strictly subcubic

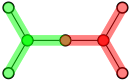
- Simple algorithm: extract a ⅄ at each step as long as possible;
- Two cases based on the degree of each vertex v:

⅄-DECOMPOSITION, strictly subcubic

- Simple algorithm: extract a ⅄ at each step as long as possible;
- Two cases based on the degree of each vertex v:
  1. degree 1: then v must be a leaf of a ⅄;

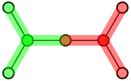
⅄-DECOMPOSITION, strictly subcubic

- Simple algorithm: extract a ⅄ at each step as long as possible;
- Two cases based on the degree of each vertex v:
  1. degree 1: then v must be a leaf of a ⅄;
  2. degree 2: then v must be the meeting point of two ⅄'s;



⅄-DECOMPOSITION, strictly subcubic

- Simple algorithm: extract a ⅄ at each step as long as possible;
- Two cases based on the degree of each vertex v:
  1. degree 1: then v must be a leaf of a ⅄;
  2. degree 2: then v must be the meeting point of two ⅄'s;



- When the algorithm stops, either G has no edge left and we have a ⅄ -decomposition, or G does not admit one.

{⅄, △}-DECOMPOSITION, strictly subcubic

- Similar approach to the ⅄-only case: we have three cases based on the degree of each vertex v:

{⅄, △}-DECOMPOSITION, strictly subcubic

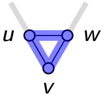
- Similar approach to the ⅄-only case: we have three cases based on the degree of each vertex v:
  1. degree 1: then v must be a leaf of a ⅄;

{⅄, △}-DECOMPOSITION, strictly subcubic

- Similar approach to the ⅄-only case: we have three cases based on the degree of each vertex v:
  1. degree 1: then v must be a leaf of a ⅄;
  2. degree 2: then let's consider v's two neighbours (u and w):

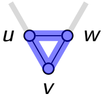
{⅄, △}-DECOMPOSITION, strictly subcubic

- Similar approach to the ⅄-only case: we have three cases based on the degree of each vertex v:
  1. degree 1: then v must be a leaf of a ⅄;
  2. degree 2: then let's consider v's two neighbours (u and w):
     - if u and w are adjacent, then we must extract the △ that u, v and w induce;

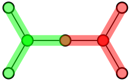


{⅄, △}-DECOMPOSITION, strictly subcubic

- Similar approach to the ⅄-only case: we have three cases based on the degree of each vertex v:
  1. degree 1: then v must be a leaf of a ⅄;
  2. degree 2: then let's consider v's two neighbours (u and w):
     - if u and w are adjacent, then we must extract the △ that u, v and w induce;

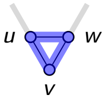


     - otherwise v is again the meeting point of two ⅄'s;

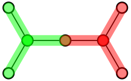


{⅄, △}-DECOMPOSITION, strictly subcubic

- Similar approach to the ⅄-only case: we have three cases based on the degree of each vertex *v*:
  1. degree 1: then *v* must be a leaf of a ⅄;
  2. degree 2: then let's consider *v*'s two neighbours (*u* and *w*):
     - if *u* and *w* are adjacent, then we must extract the △ that *u*, *v* and *w* induce;



     - otherwise *v* is again the meeting point of two ⅄'s;



- When the algorithm stops, either *G* has no edge left and we have a {⅄, △}-decomposition, or *G* does not admit one.

∘-∘-∘-∘ -DECOMPOSITION, strictly subcubic

- We now show that ∘-∘-∘-∘ -DECOMPOSITION for strictly subcubic graphs is NP-complete;

∘-∘-∘-∘ -DECOMPOSITION, strictly subcubic

- We now show that ∘-∘-∘-∘ -DECOMPOSITION for strictly subcubic graphs is NP-complete;
- We reduce from the following well-known problem:

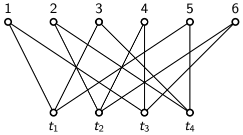
EXACT COVER BY 3-SETS (X3C)
Input: a set W and a set of triplets $T \subseteq W^3$
Question: is there a subset $T' \subseteq T$ which contains all elements of W exactly once?

∘–∘–∘–∘ -DECOMPOSITION, strictly subcubic

- We now show that ∘–∘–∘–∘ -DECOMPOSITION for strictly subcubic graphs is NP-complete;
- We reduce from the following well-known problem:

EXACT COVER BY 3-SETS (X3C)
Input: a set W and a set of triplets $T \subseteq W^3$
Question: is there a subset $T' \subseteq T$ which contains all elements of W exactly once?

Example

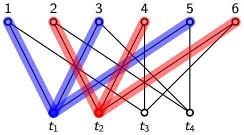


○–○–○–○ -DECOMPOSITION, strictly subcubic

- We now show that ○–○–○–○ -DECOMPOSITION for strictly subcubic graphs is NP-complete;
- We reduce from the following well-known problem:

EXACT COVER BY 3-SETS (X3C)
Input: a set W and a set of triplets $T \subseteq W^3$
Question: is there a subset $T' \subseteq T$ which contains all elements of W exactly once?

Example

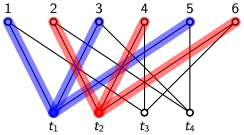


∘-∘-∘-∘ -DECOMPOSITION, strictly subcubic

- We now show that ∘-∘-∘-∘ -DECOMPOSITION for strictly subcubic graphs is NP-complete;
- We reduce from the following well-known problem:

EXACT COVER BY 3-SETS (X3C)
Input: a set W and a set of triplets $T \subseteq W^3$
Question: is there a subset $T' \subseteq T$ which contains all elements of W exactly once?

Example



- X3C remains NP-complete if the bipartite instance graph G is planar and if $deg(w) \in \{2, 3\}\ \forall\ w \in W$;

∘–∘–∘–∘ -DECOMPOSITION, strictly subcubic

The usual steps in a reduction are:

1. transforming instances of hard problem A to target problem B;

∘-∘-∘-∘ -DECOMPOSITION, strictly subcubic

The usual steps in a reduction are:

1. transforming instances of hard problem A to target problem B;
2. showing how to convert solutions to A into solutions to B;

∘–∘–∘–∘ -DECOMPOSITION, strictly subcubic

The usual steps in a reduction are:

1. transforming instances of hard problem A to target problem B;
2. showing how to convert solutions to A into solutions to B;
3. showing how to convert solutions to B into solutions to A;

-DECOMPOSITION, strictly subcubic

The usual steps in a reduction are:

1. transforming instances of hard problem A to target problem B;
2. showing how to convert solutions to A into solutions to B;
3. showing how to convert solutions to B into solutions to A;

We assume the instance to X3C is a planar bipartite graph $G = (W \cup T, E)$ with $deg(w) \in \{2, 3\} \; \forall \; w \in W$; so, we must:

∘–∘–∘–∘ -DECOMPOSITION, strictly subcubic

The usual steps in a reduction are:

1. transforming instances of hard problem A to target problem B;
2. showing how to convert solutions to A into solutions to B;
3. showing how to convert solutions to B into solutions to A;

We assume the instance to X3C is a planar bipartite graph $G = (W \cup T, E)$ with $deg(w) \in \{2, 3\} \ \forall \ w \in W$; so, we must:

1. transform G into a graph G' to decompose;

∘–∘–∘–∘ -DECOMPOSITION, strictly subcubic

The usual steps in a reduction are:

1. transforming instances of hard problem A to target problem B;
2. showing how to convert solutions to A into solutions to B;
3. showing how to convert solutions to B into solutions to A;

We assume the instance to X3C is a planar bipartite graph $G = (W \cup T, E)$ with $deg(w) \in \{2,3\} \; \forall \; w \in W$; so, we must:

1. transform G into a graph G' to decompose;
2. convert triplet selections for G into ∘–∘–∘–∘ -decompositions for G';

∘-∘-∘-∘ -DECOMPOSITION, strictly subcubic

The usual steps in a reduction are:

1. transforming instances of hard problem A to target problem B;
2. showing how to convert solutions to A into solutions to B;
3. showing how to convert solutions to B into solutions to A;

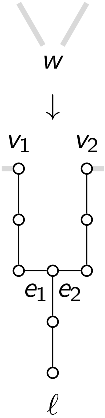
We assume the instance to X3C is a planar bipartite graph $G = (W \cup T, E)$ with $deg(w) \in \{2, 3\} \ \forall \ w \in W$; so, we must:

1. transform G into a graph G' to decompose;
2. convert triplet selections for G into ∘-∘-∘-∘ -decompositions for G';
3. convert ∘-∘-∘-∘ -decompositions for G' into triplet selections for G;

Reducing from PLANAR X3C 1/3: transformation

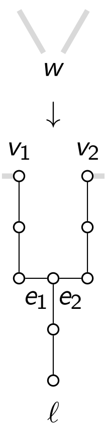
Reducing from PLANAR X3C 1/3: transformation

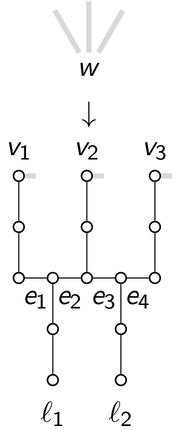
degree-2 elements



Reducing from PLANAR X3C 1/3: transformation

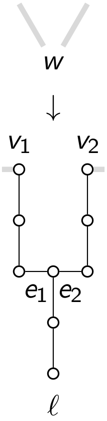
degree-2 elements | degree-3 elements



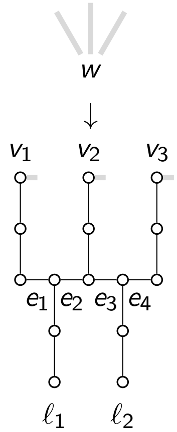


Reducing from PLANAR X3C 1/3: transformation

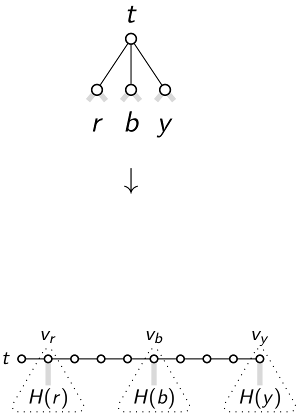
degree-2 elements



degree-3 elements



triplets

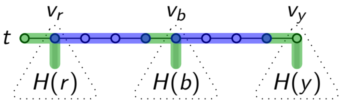


Reducing from PLANAR X3C 2/3: converting selections

We express the (un)selection of a triplet using suitable ∘-∘-∘-∘'s:

Reducing from PLANAR X3C 2/3: converting selections

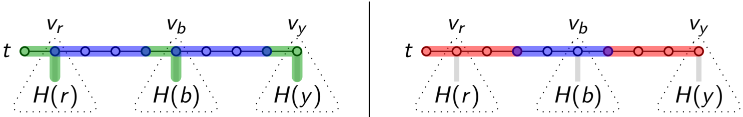
We express the (un)selection of a triplet using suitable ∘–∘–∘–∘'s:



selected

Reducing from PLANAR X3C 2/3: converting selections

We express the (un)selection of a triplet using suitable ○–○–○–○'s:

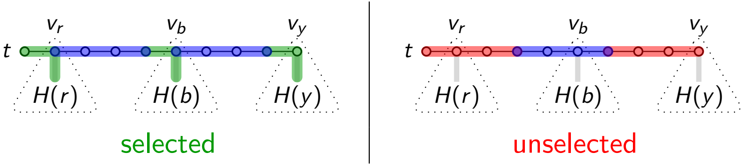


selected

unselected

Reducing from PLANAR X3C 2/3: converting selections

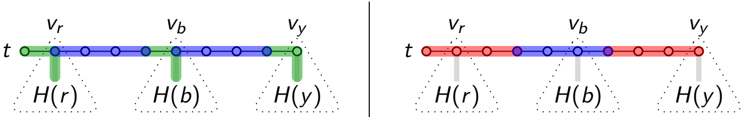
We express the (un)selection of a triplet using suitable ∘–∘–∘–∘'s:



Each element is selected by exactly one selected triplet; green paths uniquely determine the rest of the decomposition:

Reducing from PLANAR X3C 2/3: converting selections

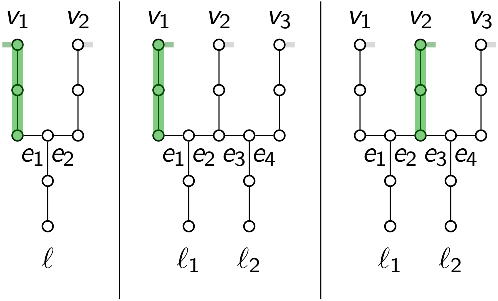
We express the (un)selection of a triplet using suitable ○-○-○-○'s:



selected

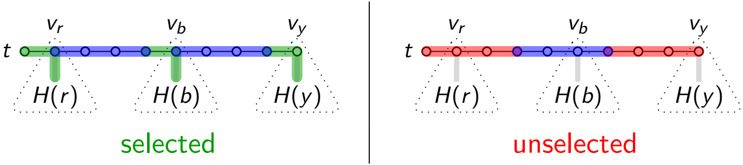
unselected

Each element is selected by exactly one selected triplet; green paths uniquely determine the rest of the decomposition:

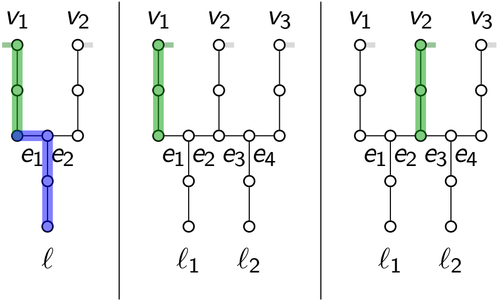


Reducing from PLANAR X3C 2/3: converting selections

We express the (un)selection of a triplet using suitable ∘-∘-∘-∘'s:

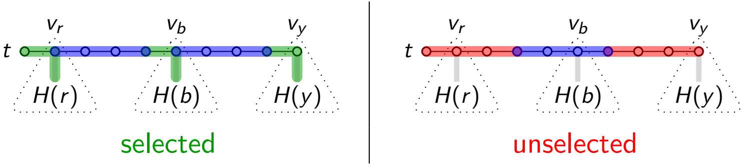


Each element is selected by exactly one selected triplet; green paths uniquely determine the rest of the decomposition:

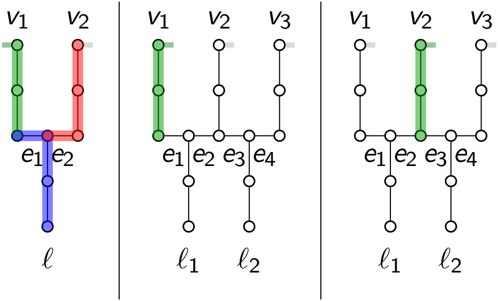


Reducing from PLANAR X3C 2/3: converting selections

We express the (un)selection of a triplet using suitable ∘-∘-∘-∘'s:

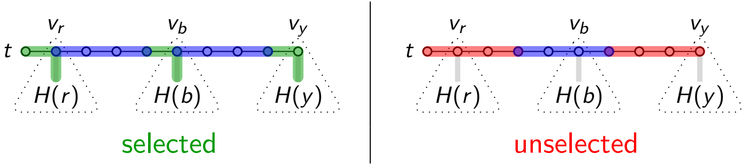


Each element is selected by exactly one selected triplet; green paths uniquely determine the rest of the decomposition:



Reducing from PLANAR X3C 2/3: converting selections

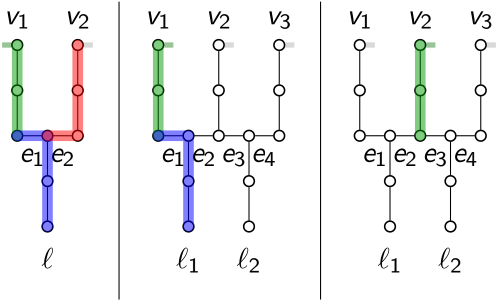
We express the (un)selection of a triplet using suitable ∘–∘–∘–∘'s:



selected

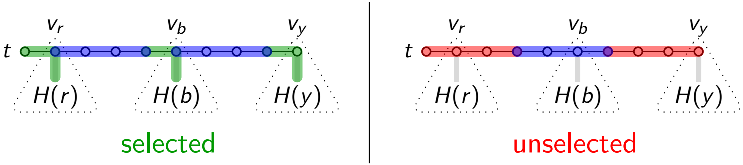
unselected

Each element is selected by exactly one selected triplet; green paths uniquely determine the rest of the decomposition:

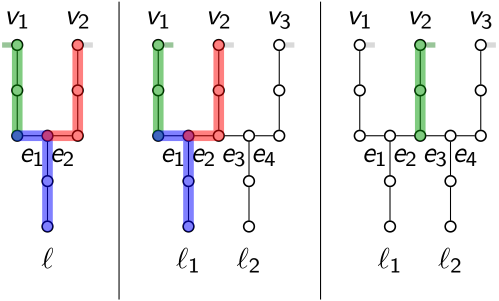


Reducing from PLANAR X3C 2/3: converting selections

We express the (un)selection of a triplet using suitable ○-○-○-○'s:

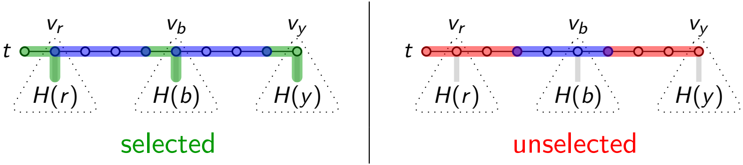


Each element is selected by exactly one selected triplet; green paths uniquely determine the rest of the decomposition:

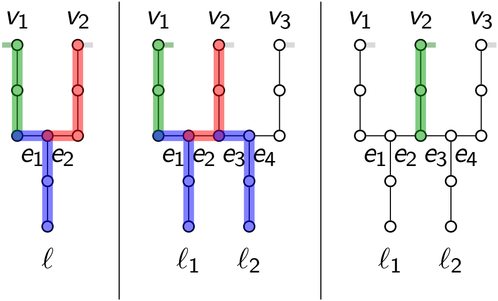


Reducing from PLANAR X3C 2/3: converting selections

We express the (un)selection of a triplet using suitable ∘-∘-∘-∘'s:

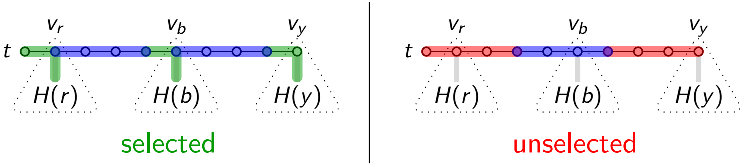


Each element is selected by exactly one selected triplet; green paths uniquely determine the rest of the decomposition:

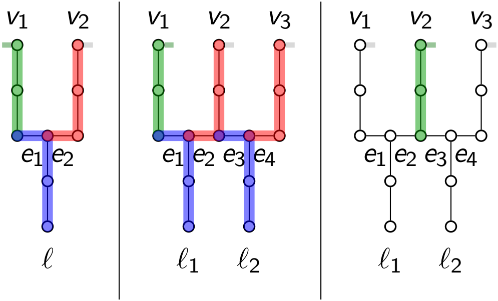


Reducing from PLANAR X3C 2/3: converting selections

We express the (un)selection of a triplet using suitable ∘-∘-∘-∘'s:

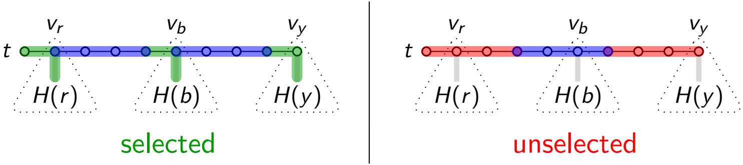


Each element is selected by exactly one selected triplet; green paths uniquely determine the rest of the decomposition:

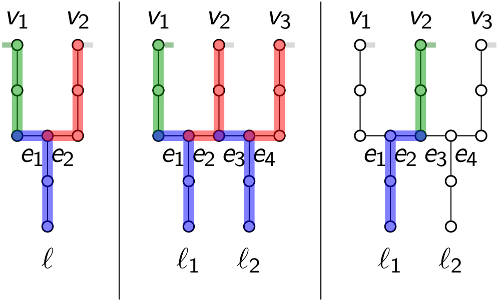


Reducing from PLANAR X3C 2/3: converting selections

We express the (un)selection of a triplet using suitable ∘–∘–∘–∘'s:

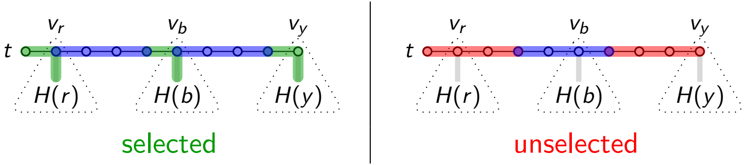


Each element is selected by exactly one selected triplet; green paths uniquely determine the rest of the decomposition:



Reducing from PLANAR X3C 2/3: converting selections

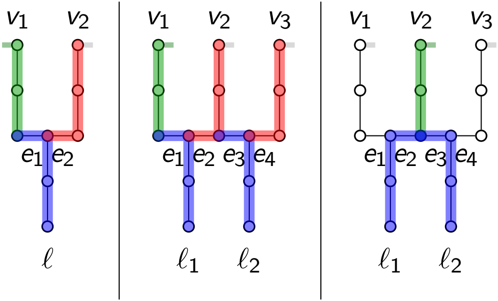
We express the (un)selection of a triplet using suitable ○–○–○–○'s:



selected

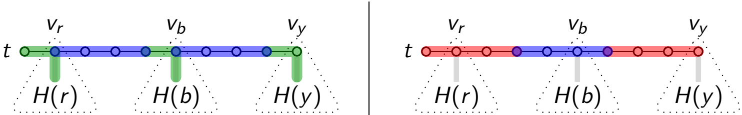
unselected

Each element is selected by exactly one selected triplet; green paths uniquely determine the rest of the decomposition:



Reducing from PLANAR X3C 2/3: converting selections

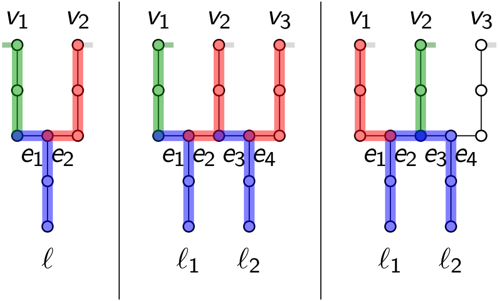
We express the (un)selection of a triplet using suitable ∘-∘-∘-∘'s:



selected

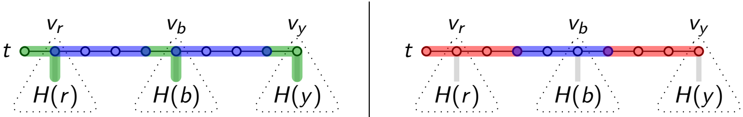
unselected

Each element is selected by exactly one selected triplet; green paths uniquely determine the rest of the decomposition:



Reducing from PLANAR X3C 2/3: converting selections

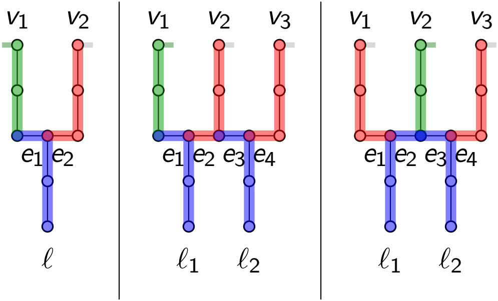
We express the (un)selection of a triplet using suitable ∘–∘–∘–∘'s:



selected

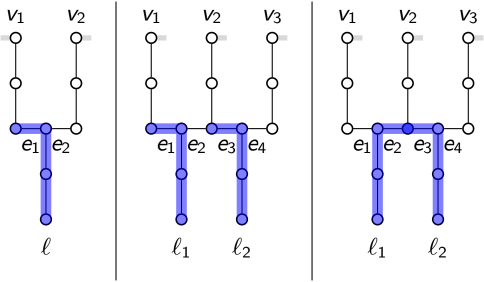
unselected

Each element is selected by exactly one selected triplet; green paths uniquely determine the rest of the decomposition:



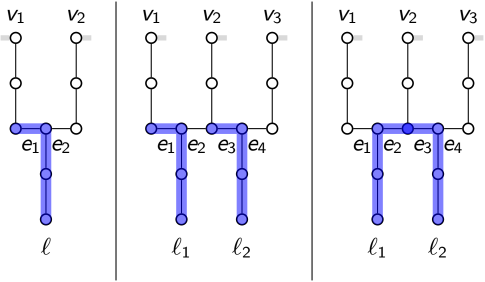
Reducing from PLANAR X3C 3/3: converting decompositions

Start from the leaves of element gadgets and propagate implications:

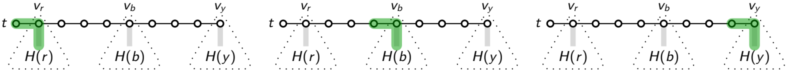


Reducing from PLANAR X3C 3/3: converting decompositions

Start from the leaves of element gadgets and propagate implications:

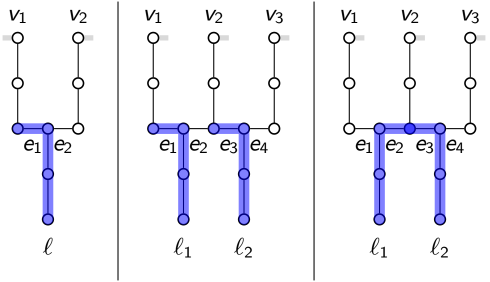


Every element gadget ends up with exactly one green path and is therefore selected. In turn, the green edge coming out of the element forces the selection of the other two elements in the triplet gadget that ends up selecting it:

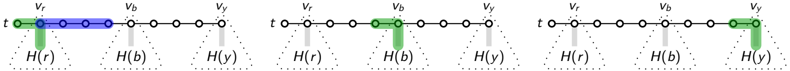


Reducing from PLANAR X3C 3/3: converting decompositions

Start from the leaves of element gadgets and propagate implications:

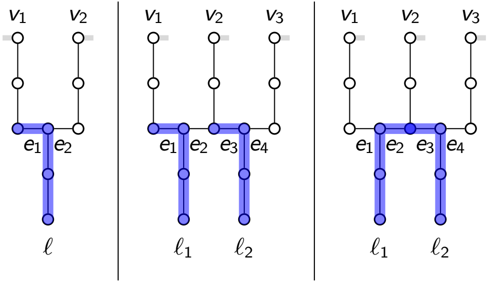


Every element gadget ends up with exactly one green path and is therefore selected. In turn, the green edge coming out of the element forces the selection of the other two elements in the triplet gadget that ends up selecting it:

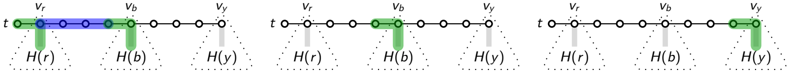


Reducing from PLANAR X3C 3/3: converting decompositions

Start from the leaves of element gadgets and propagate implications:

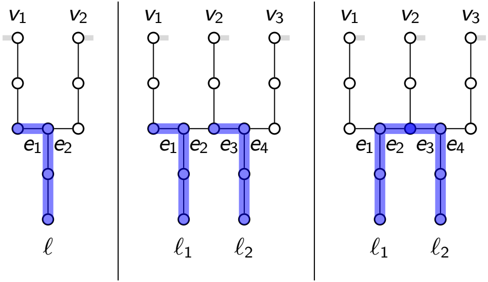


Every element gadget ends up with exactly one green path and is therefore selected. In turn, the green edge coming out of the element forces the selection of the other two elements in the triplet gadget that ends up selecting it:

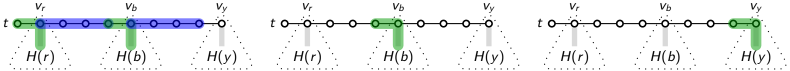


Reducing from PLANAR X3C 3/3: converting decompositions

Start from the leaves of element gadgets and propagate implications:

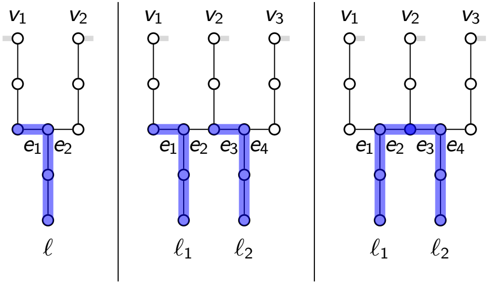


Every element gadget ends up with exactly one green path and is therefore selected. In turn, the green edge coming out of the element forces the selection of the other two elements in the triplet gadget that ends up selecting it:

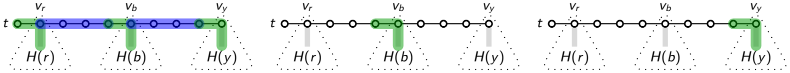


Reducing from PLANAR X3C 3/3: converting decompositions

Start from the leaves of element gadgets and propagate implications:

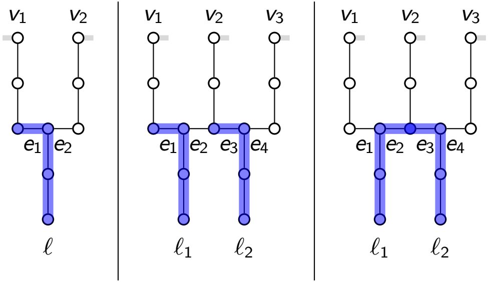


Every element gadget ends up with exactly one green path and is therefore selected. In turn, the green edge coming out of the element forces the selection of the other two elements in the triplet gadget that ends up selecting it:

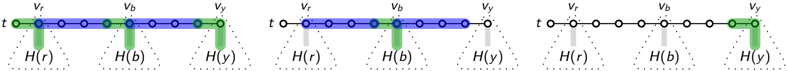


Reducing from PLANAR X3C 3/3: converting decompositions

Start from the leaves of element gadgets and propagate implications:

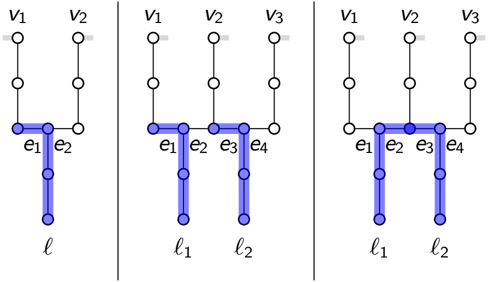


Every element gadget ends up with exactly one green path and is therefore selected. In turn, the green edge coming out of the element forces the selection of the other two elements in the triplet gadget that ends up selecting it:

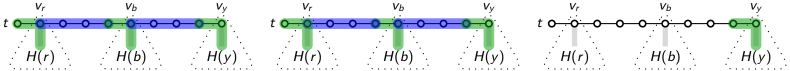


Reducing from PLANAR X3C 3/3: converting decompositions

Start from the leaves of element gadgets and propagate implications:

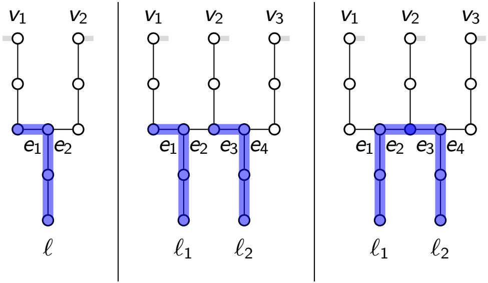


Every element gadget ends up with exactly one green path and is therefore selected. In turn, the green edge coming out of the element forces the selection of the other two elements in the triplet gadget that ends up selecting it:

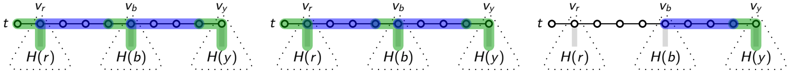


Reducing from PLANAR X3C 3/3: converting decompositions

Start from the leaves of element gadgets and propagate implications:

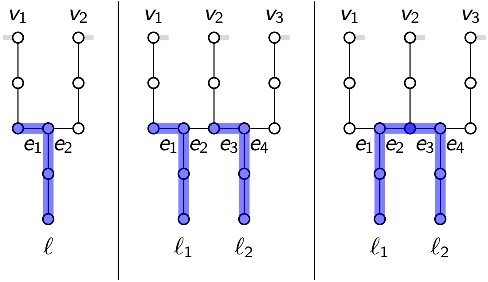


Every element gadget ends up with exactly one green path and is therefore selected. In turn, the green edge coming out of the element forces the selection of the other two elements in the triplet gadget that ends up selecting it:

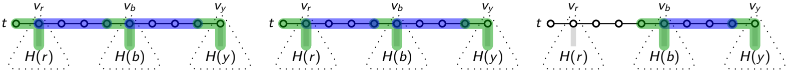


Reducing from PLANAR X3C 3/3: converting decompositions

Start from the leaves of element gadgets and propagate implications:

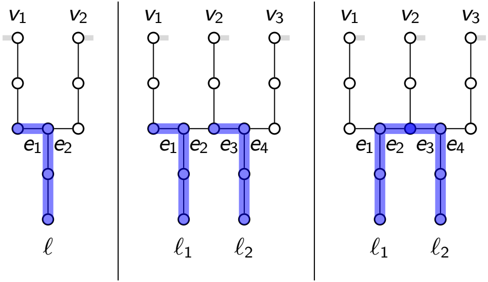


Every element gadget ends up with exactly one green path and is therefore selected. In turn, the green edge coming out of the element forces the selection of the other two elements in the triplet gadget that ends up selecting it:

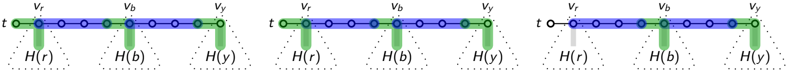


Reducing from PLANAR X3C 3/3: converting decompositions

Start from the leaves of element gadgets and propagate implications:

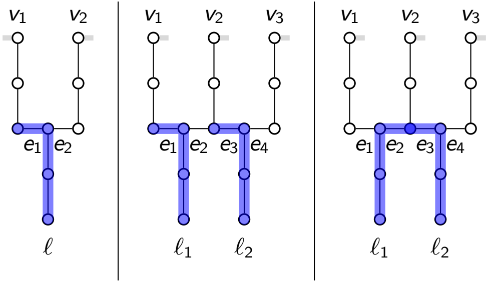


Every element gadget ends up with exactly one green path and is therefore selected. In turn, the green edge coming out of the element forces the selection of the other two elements in the triplet gadget that ends up selecting it:

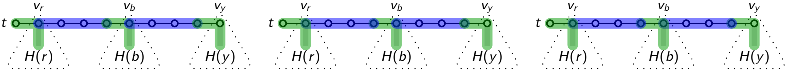


Reducing from PLANAR X3C 3/3: converting decompositions

Start from the leaves of element gadgets and propagate implications:



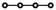
Every element gadget ends up with exactly one green path and is therefore selected. In turn, the green edge coming out of the element forces the selection of the other two elements in the triplet gadget that ends up selecting it:



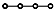
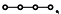
Wrapping up hardness results for strictly subcubic graphs

- In the strictly subcubic case, two reductions prove the hardness of :

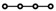
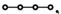
Wrapping up hardness results for strictly subcubic graphs

- In the strictly subcubic case, two reductions prove the hardness of :
  1.  -DECOMPOSITION (just shown);

Wrapping up hardness results for strictly subcubic graphs

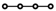
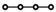
- In the strictly subcubic case, two reductions prove the hardness of :
  1.  -DECOMPOSITION (just shown);
  2. { , }-DECOMPOSITION (skipped but similar);

Wrapping up hardness results for strictly subcubic graphs

- In the strictly subcubic case, two reductions prove the hardness of :
  1.  -DECOMPOSITION (just shown);
  2. { , }-DECOMPOSITION (skipped but similar);

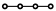
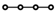
- The transformed graphs are planar and bipartite

Wrapping up hardness results for strictly subcubic graphs

- In the strictly subcubic case, two reductions prove the hardness of :
  1.  -DECOMPOSITION (just shown);
  2. { , }-DECOMPOSITION (skipped but similar);

- The transformed graphs are planar and bipartite ⇒ no odd cycles

Wrapping up hardness results for strictly subcubic graphs

- In the strictly subcubic case, two reductions prove the hardness of :
  1.  -DECOMPOSITION (just shown);
  2. {, }-DECOMPOSITION (skipped but similar);

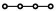
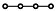
- The transformed graphs are planar and bipartite ⇒ no odd cycles ⇒ no .

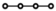
Wrapping up hardness results for strictly subcubic graphs

- In the strictly subcubic case, two reductions prove the hardness of :
  1. ∘-∘-∘-∘ -DECOMPOSITION (just shown);
  2. {∘-∘-∘-∘, ⅄}-DECOMPOSITION (skipped but similar);

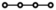
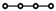
- The transformed graphs are planar and bipartite ⇒ no odd cycles ⇒ no △. So we immediately obtain the hardness of:

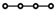
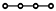
Wrapping up hardness results for strictly subcubic graphs

- In the strictly subcubic case, two reductions prove the hardness of :
  1.  -DECOMPOSITION (just shown);
  2. { , }-DECOMPOSITION (skipped but similar);

- The transformed graphs are planar and bipartite ⇒ no odd cycles ⇒ no . So we immediately obtain the hardness of:
  1. { , }-DECOMPOSITION;

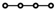
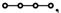
Wrapping up hardness results for strictly subcubic graphs

- In the strictly subcubic case, two reductions prove the hardness of :
  1.  -DECOMPOSITION (just shown);
  2. $\{$ , $\}$-DECOMPOSITION (skipped but similar);

- The transformed graphs are planar and bipartite ⇒ no odd cycles ⇒ no . So we immediately obtain the hardness of:
  1. $\{$ , $\}$-DECOMPOSITION;
  2. $\{$ , , $\}$-DECOMPOSITION.

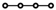
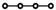
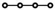
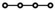
The cubic case

We now move on to the cubic case, i.e. every vertex of G has degree 3.

-  -DECOMPOSITION and { , }-DECOMPOSITION become easy!

The cubic case

We now move on to the cubic case, i.e. every vertex of G has degree 3.

-  -DECOMPOSITION and { , }-DECOMPOSITION become easy!
- { , }-DECOMPOSITION and { , , }-DECOMPOSITION remain hard, but we need new reductions;

The cubic case

We now move on to the cubic case, i.e. every vertex of G has degree 3.

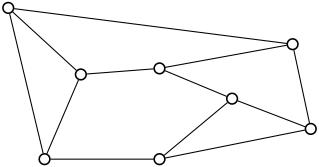
- ∘–∘–∘–∘ -DECOMPOSITION and {∘–∘–∘–∘, △}-DECOMPOSITION become easy!
- {∘–∘–∘–∘, ⅄}-DECOMPOSITION and {∘–∘–∘–∘, ⅄, △}-DECOMPOSITION remain hard, but we need new reductions;
- △ -DECOMPOSITION remains trivial (never possible in the cubic case);

∘–∘–∘–∘ -DECOMPOSITION, cubic

We need the following result:

Proposition ([6])

A cubic graph admits a ∘–∘–∘–∘ -decomposition if and only if it has a perfect matching.

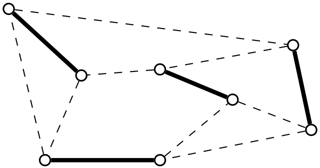


∘–∘–∘–∘ -DECOMPOSITION, cubic

We need the following result:

Proposition ([6])

A cubic graph admits a ∘–∘–∘–∘ -decomposition if and only if it has a perfect matching.

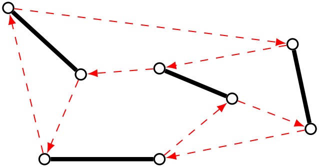


∘–∘–∘–∘ -DECOMPOSITION, cubic

We need the following result:

Proposition ([6])

A cubic graph admits a ∘–∘–∘–∘ *-decomposition if and only if it has a perfect matching.*

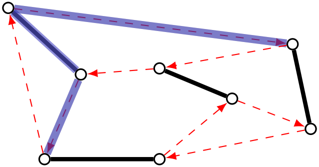


∘–∘–∘–∘ -DECOMPOSITION, cubic

We need the following result:

Proposition ([6])

A cubic graph admits a ∘–∘–∘–∘ -decomposition if and only if it has a perfect matching.

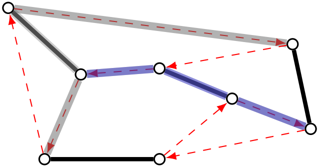


∘–∘–∘–∘ -DECOMPOSITION, cubic

We need the following result:

Proposition ([6])

A cubic graph admits a ∘–∘–∘–∘ -decomposition if and only if it has a perfect matching.

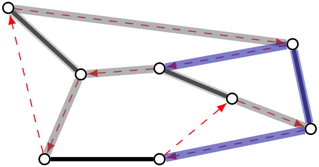


∘–∘–∘–∘ -DECOMPOSITION, cubic

We need the following result:

Proposition ([6])

A cubic graph admits a ∘–∘–∘–∘ -decomposition if and only if it has a perfect matching.

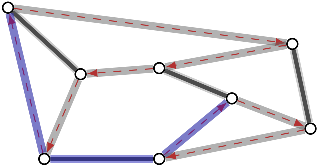


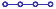
∘-∘-∘-∘ -DECOMPOSITION, cubic

We need the following result:

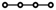
Proposition ([6])

A cubic graph admits a ∘-∘-∘-∘ -decomposition if and only if it has a perfect matching.



 -DECOMPOSITION, cubic

Proposition

A cubic graph admits a { , }*-decomposition if and only if it has a perfect matching.*

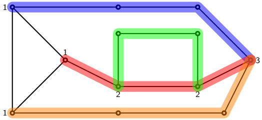
∘-∘-∘-∘ -DECOMPOSITION, cubic

Proposition

A cubic graph admits a $\{$△, ∘-∘-∘-∘$\}$*-decomposition if and only if it has a perfect matching.*

Proof.

Each vertex in V is covered by k ∘-∘-∘-∘ 's ($k \in \{1, 2, 3\}$). Example:



$\Rightarrow V = V_1 \cup V_2 \cup V_3.$

□

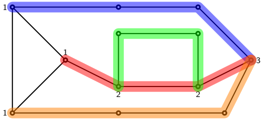
∘-∘-∘-∘ -DECOMPOSITION, cubic

Proposition

A cubic graph admits a {△, ∘-∘-∘-∘}*-decomposition if and only if it has a perfect matching.*

Proof.

Each vertex in V is covered by k ∘-∘-∘-∘ 's ($k \in \{1, 2, 3\}$). Example:



⇒ $V = V_1 \cup V_2 \cup V_3$. Let's compute the number p of ∘-∘-∘-∘ 's in a decomposition; we have (details omitted):

$$(3|V_3| + |V_2| + |V_1|)/2 = p = |V_2|/2$$

□

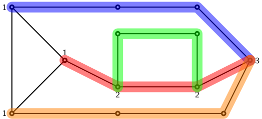
∘–∘–∘–∘ -DECOMPOSITION, cubic

Proposition

A cubic graph admits a {△, ∘–∘–∘–∘}*-decomposition if and only if it has a perfect matching.*

Proof.

Each vertex in V is covered by k ∘–∘–∘–∘ 's ($k \in \{1, 2, 3\}$). Example:



$\Rightarrow V = V_1 \cup V_2 \cup V_3$. Let's compute the number p of ∘–∘–∘–∘ 's in a decomposition; we have (details omitted):

$$(3|V_3| + |V_2| + |V_1|)/2 = p = |V_2|/2$$

So $V_1 = V_3 = \emptyset$; and since V_1 is the set of vertices that belong to a △ , no decomposition with a △ exists. □

⅄ -DECOMPOSITION, cubic

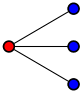
We obtain a simple characterisation of ⅄ -decomposable cubic graphs:

Proposition

A cubic graph admits a ⅄ -decomposition if and only if it is bipartite.

Proof.

$\boxed{\Rightarrow}$



A center (red) belongs to only one subgraph

⇒ Bipartition: centers – leaves

(each edge connects a center and a leaf)

⅄ -DECOMPOSITION, cubic

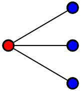
We obtain a simple characterisation of ⅄ -decomposable cubic graphs:

Proposition

A cubic graph admits a ⅄ -decomposition if and only if it is bipartite.

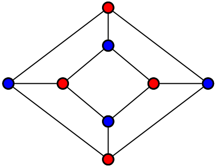
Proof.

$\boxed{\Rightarrow}$



A center (red) belongs to only one subgraph
⇒ Bipartition: centers – leaves
(each edge connects a center and a leaf)

$\boxed{\Leftarrow}$



Use one part for centers, the other for leaves

□

⅄ -DECOMPOSITION, cubic

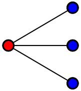
We obtain a simple characterisation of ⅄ -decomposable cubic graphs:

Proposition

A cubic graph admits a ⅄ -decomposition if and only if it is bipartite.

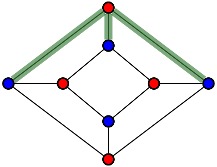
Proof.

$\boxed{\Rightarrow}$



A center (red) belongs to only one subgraph
⇒ Bipartition: centers – leaves
(each edge connects a center and a leaf)

$\boxed{\Leftarrow}$



Use one part for centers, the other for leaves

□

⅄ -DECOMPOSITION, cubic

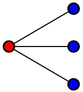
We obtain a simple characterisation of ⅄ -decomposable cubic graphs:

Proposition

A cubic graph admits a ⅄ -decomposition if and only if it is bipartite.

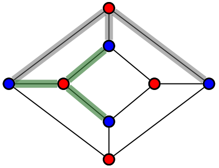
Proof.

$\boxed{\Rightarrow}$



A center (red) belongs to only one subgraph
⇒ Bipartition: centers – leaves
(each edge connects a center and a leaf)

$\boxed{\Leftarrow}$



Use one part for centers, the other for leaves

□

⅄ -DECOMPOSITION, cubic

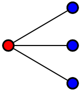
We obtain a simple characterisation of ⅄ -decomposable cubic graphs:

Proposition

A cubic graph admits a ⅄ -decomposition if and only if it is bipartite.

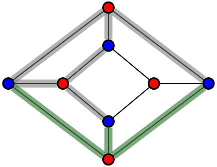
Proof.

$\boxed{\Rightarrow}$



A center (red) belongs to only one subgraph
⇒ Bipartition: centers – leaves
(each edge connects a center and a leaf)

$\boxed{\Leftarrow}$



Use one part for centers, the other for leaves

□

⅄ -DECOMPOSITION, cubic

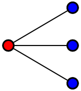
We obtain a simple characterisation of ⅄ -decomposable cubic graphs:

Proposition

A cubic graph admits a ⅄ -decomposition if and only if it is bipartite.

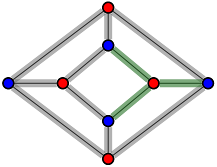
Proof.

$\boxed{\Rightarrow}$



A center (red) belongs to only one subgraph
⇒ Bipartition: centers – leaves
(each edge connects a center and a leaf)

$\boxed{\Leftarrow}$

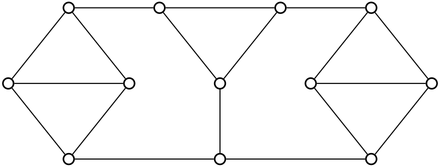


Use one part for centers, the other for leaves

□

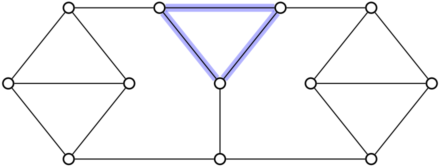
{⅄, △}-DECOMPOSITION, cubic

What if we also allow △'s?



{⅄, △}-DECOMPOSITION, cubic

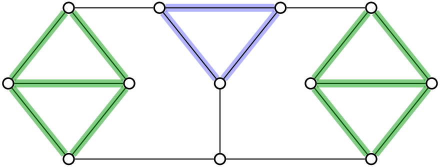
What if we also allow △'s?



We distinguish between *isolated* and *nonisolated* triangles:

{⅄, △}-DECOMPOSITION, cubic

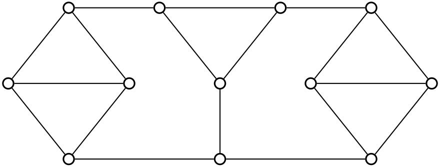
What if we also allow △'s?



We distinguish between *isolated* and *nonisolated* triangles:

{⅄, △}-DECOMPOSITION, cubic

What if we also allow △'s?



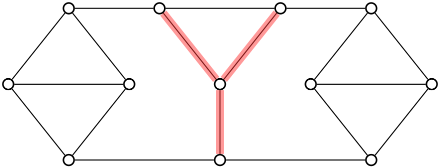
We distinguish between *isolated* and *nonisolated* triangles:

Lemma

If a cubic graph G admits a {⅄, △}-decomposition D, then every isolated △ in G belongs to D.

{⅄, △}-DECOMPOSITION, cubic

What if we also allow △'s?



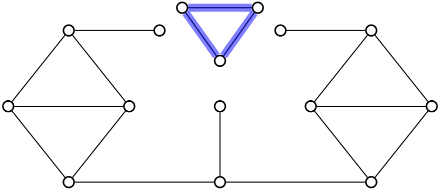
We distinguish between *isolated* and *nonisolated* triangles:

Lemma

If a cubic graph G admits a {⅄, △}-decomposition D, then every isolated △ in G belongs to D.

{⅄, △}-DECOMPOSITION, cubic

What if we also allow △'s?



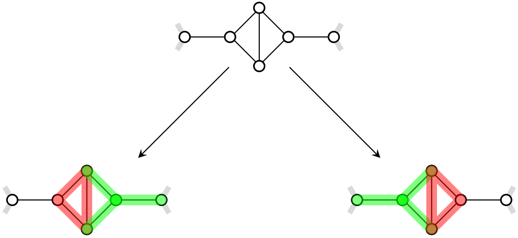
We distinguish between *isolated* and *nonisolated* triangles:

Lemma

If a cubic graph G admits a {⅄, △}-decomposition D, then every isolated △ in G belongs to D.

{⅄, △}-DECOMPOSITION, cubic

If G also contains nonisolated △'s, then we only have two choices to try:



{⅄, △}-DECOMPOSITION, cubic

The algorithm proceeds as follows:

1. extract all isolated triangles and add them to the decomposition;

{⅄, △}-DECOMPOSITION, cubic

The algorithm proceeds as follows:

1. extract all isolated triangles and add them to the decomposition;
2. if there's a diamond, try either option for the decomposition;

{⅄, △}-DECOMPOSITION, cubic

The algorithm proceeds as follows:

1. extract all isolated triangles and add them to the decomposition;
2. if there's a diamond, try either option for the decomposition;
3. if the resulting graph is still cubic, find a ⅄ -decomposition using the previous algorithm;

{⅄, △}-DECOMPOSITION, cubic

The algorithm proceeds as follows:

1. extract all isolated triangles and add them to the decomposition;
2. if there's a diamond, try either option for the decomposition;
3. if the resulting graph is still cubic, find a ⅄ -decomposition using the previous algorithm;
4. otherwise, run the {⅄, △}-decomposition algorithm for strictly subcubic graphs;

{⅄, ∘-∘-∘-∘}-DECOMPOSITION, cubic

We now show that {⅄, ∘-∘-∘-∘}-DECOMPOSITION is NP-complete, using three reductions (I'll skip tons of details and just explain the gist of the first one):

CUBIC MONOTONE 1-IN-3 SATISFIABILITY

$\leq_P$ DEGREE-2,3 {⅄, △, ∘-∘-∘-∘}-DECOMPOSITION WITH MARKED EDGES

$\leq_P$ {⅄, △, ∘-∘-∘-∘}-DECOMPOSITION WITH MARKED EDGES

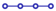
$\leq_P$ {⅄, ∘-∘-∘-∘}-DECOMPOSITION

{⅄, ∘-∘-∘-∘}-DECOMPOSITION, cubic

We now show that {⅄, ∘-∘-∘-∘}-DECOMPOSITION is NP-complete, using three reductions (I'll skip tons of details and just explain the gist of the first one):

CUBIC MONOTONE 1-IN-3 SATISFIABILITY
$\leq_P$ DEGREE-2,3 {⅄, △, ∘-∘-∘-∘}-DECOMPOSITION WITH MARKED EDGES
$\leq_P$ {⅄, △, ∘-∘-∘-∘}-DECOMPOSITION WITH MARKED EDGES
$\leq_P$ {⅄, ∘-∘-∘-∘}-DECOMPOSITION

A similar approach can be used to show the NP-completeness of {⅄, △, ∘-∘-∘-∘}-DECOMPOSITION.

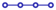
{ , }-DECOMPOSITION, cubic

We reduce from the following NP-complete problem:

SAT(ISFIABILITY)

Input: a Boolean formula $\phi = C_1 \wedge C_2 \wedge \cdots$;

Question: is there an assignment $f : \Sigma \to \{\text{TRUE}, \text{FALSE}\}$ such that each clause C_i contains one TRUE literal?

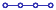
{ , }-DECOMPOSITION, cubic

We reduce from the following NP-complete problem:

MONOTONE SAT(ISFIABILITY)

Input: a Boolean formula $\phi = C_1 \wedge C_2 \wedge \cdots$ without negations;

Question: is there an assignment $f : \Sigma \to \{\text{TRUE}, \text{FALSE}\}$ such that each clause C_i contains one TRUE literal?

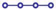
{ , }-DECOMPOSITION, cubic

We reduce from the following NP-complete problem:

MONOTONE 1-IN-3 SAT(ISFIABILITY)

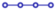
Input: a Boolean formula $\phi = C_1 \wedge C_2 \wedge \cdots$ without negations; $|C_i| = 3$ for each i

Question: is there an assignment $f : \Sigma \rightarrow \{\text{TRUE}, \text{FALSE}\}$ such that each clause C_i contains exactly one TRUE literal?

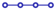
{ , }-DECOMPOSITION, cubic

We reduce from the following NP-complete problem:

CUBIC MONOTONE 1-IN-3 SAT(ISFIABILITY)
Input: a Boolean formula $\phi = C_1 \wedge C_2 \wedge \cdots$ without negations; $|C_i| = 3$ for each i and each literal appears in exactly three clauses;
Question: is there an assignment $f : \Sigma \rightarrow \{\text{TRUE}, \text{FALSE}\}$ such that each clause C_i contains exactly one TRUE literal?

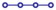
{ , }-DECOMPOSITION, cubic

Echoing the steps of the previous reduction, we assume the instance to . . . SAT is a bipartite cubic graph G; so, we must:

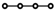
{ , }-DECOMPOSITION, cubic

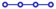
Echoing the steps of the previous reduction, we assume the instance to . . . SAT is a bipartite cubic graph G; so, we must:

1. transform G into a graph G' to decompose;

{ , }-DECOMPOSITION, cubic

Echoing the steps of the previous reduction, we assume the instance to . . . SAT is a bipartite cubic graph G; so, we must:

1. transform G into a graph G' to decompose;
2. convert truth assignments for G into { , }-decompositions for G';

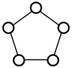
{ , }-DECOMPOSITION, cubic

Echoing the steps of the previous reduction, we assume the instance to . . . SAT is a bipartite cubic graph G; so, we must:

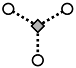
1. transform G into a graph G' to decompose;
2. convert truth assignments for G into { , }-decompositions for G';
3. convert { , }-decompositions for G' into truth assignments for G;

The reduction from CUBIC MONO-1-IN-3-SAT

Clause



Variable



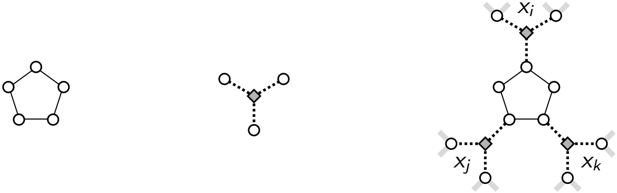
1. Map clauses onto C_5's and variables onto “marked” 's.

The reduction from CUBIC MONO-1-IN-3-SAT

Clause

Variable

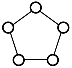
$C = x_i \vee x_j \vee x_k$



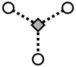
1. Map clauses onto C_5’s and variables onto “marked” ⅄’s.

The reduction from CUBIC MONO-1-IN-3-SAT

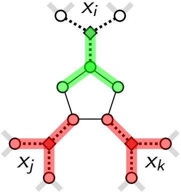
Clause



Variable

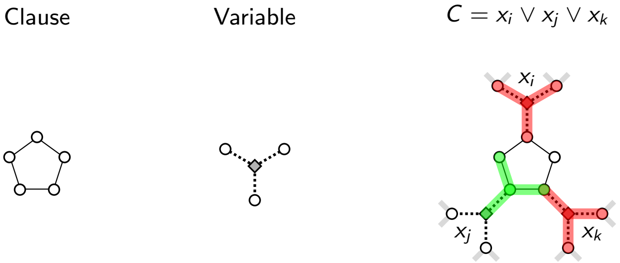


$C = x_i \vee x_j \vee x_k$



1. Map clauses onto C_5's and variables onto "marked" ⅄'s.
2. From assignments to decompositions: variables set to FALSE yield red ⅄'s, those set to TRUE yield green ⅄'s

The reduction from CUBIC MONO-1-IN-3-SAT


Clause

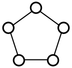
Variable

$C = x_i \vee x_j \vee x_k$

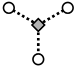
1. Map clauses onto C_5's and variables onto "marked" ⅄'s.
2. From assignments to decompositions: variables set to FALSE yield red ⅄'s, those set to TRUE yield green ⅄'s

The reduction from CUBIC MONO-1-IN-3-SAT

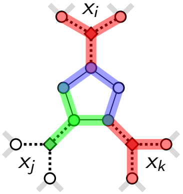
Clause



Variable



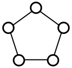
$C = x_i \vee x_j \vee x_k$



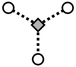
1. Map clauses onto C_5's and variables onto "marked" ⅄'s.
2. From assignments to decompositions: variables set to FALSE yield red ⅄'s, those set to TRUE yield green ⅄'s

The reduction from CUBIC MONO-1-IN-3-SAT ⅄ ∘–∘–∘–∘

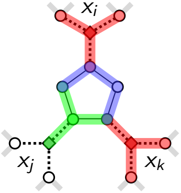
Clause



Variable



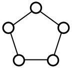
$C = x_i \vee x_j \vee x_k$



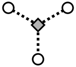
1. Map clauses onto C_5's and variables onto "marked" ⅄'s.
2. From assignments to decompositions: variables set to FALSE yield red ⅄'s, those set to TRUE yield green ⅄'s
3. From decompositions to assignments: show that a decomposable graph **must** conform to the above configuration

The reduction from CUBIC MONO-1-IN-3-SAT

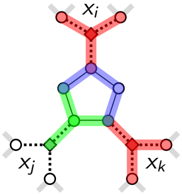
Clause



Variable



$C = x_i \lor x_j \lor x_k$

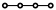
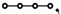


1. Map clauses onto C_5's and variables onto "marked" ⅄'s.
2. From assignments to decompositions: variables set to FALSE yield red ⅄'s, those set to TRUE yield green ⅄'s
3. From decompositions to assignments: show that a decomposable graph **must** conform to the above configuration

Marked edges are annoying and must undergo further modifications (hence the other reductions).

Encores

With (a lot) more work, we can show that

- { , }-DECOMPOSITION and
- { , , }-DECOMPOSITION

remain hard if the cubic graph is planar and -free. Ingredients:

- another variant of SAT (namely, CUBIC PLANAR MONOTONE 1-IN-3 SAT)
- another intermediate problem;
- . . . and a few more pages of reduction;

Conclusions

- We now know everything regarding S-DECOMPOSITION if G is subcubic and S is any combination of connected graphs on 3 edges.
- Possible future work:
  - what G is k-regular and $S =$ all connected subgraphs of size k for any $k > 3$?
  - do easy problems remain easy under natural generalisations? i.e.
    - P_{k+1}-DECOMPOSITION for k-regular graphs;
    - $K_{1,k}$-DECOMPOSITION for k-regular graphs;
    - ...

Thank you!

References

[1] Jeremiah Blocki and Ryan Williams.
Resolving the complexity of some data privacy problems.
In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP 2010*, , volume 6199 of *Lecture Notes in Computer Science*, pages 393–404. Springer, July 2010.

[2] Dorit Dor and Michael Tarsi.
Graph decomposition is NP-complete: A complete proof of Holyer's conjecture.
SIAM J. Comput., 26:1166–1187, 1997.

[3] M. E. Dyer and A. M. Frieze.
On the complexity of partitioning graphs into connected subgraphs.
Discrete Appl. Math., 10(2):139–153, 1985.

[4] Éric Fusy.
Transversal structures on triangulations: A combinatorial study and straight-line drawings.
Discrete Math., 309(7):1870–1894, 2009.

[5] I Holyer.
The NP-completeness of some edge-partition problems.
SIAM J. Comput., 10(4):713–717, 1981.

[6] Anton Kotzig.
Z teorie konečných pravidelných grafov tretieho a štvrtého stupňa.
Časopis pro pěstování matematiky, pages 76–92, 1957.

[7] Xavier Muñoz, Zhentao Li, and Ignasi Sau.
Edge-partitioning regular graphs for ring traffic grooming with a priori placement of the ADMs.
SIAM J. Discrete Math., 25(4):1490–1505, 2011.