Stable sets in {ISK4,wheel}-free graphs

Martin Milanič[1], Irena Penev[2], Nicolas Trotignon[3]

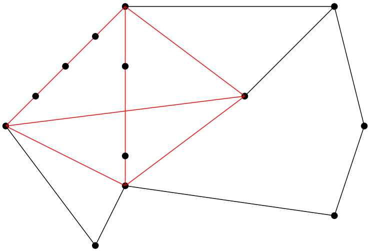
June 16, 2015

Algorithmic Graph Theory on the Adriatic Coast
Koper, Slovenia

[1]University of Primorska, Slovenia
[2]LIP, ENS de Lyon, France
[3]LIP, ENS de Lyon, France

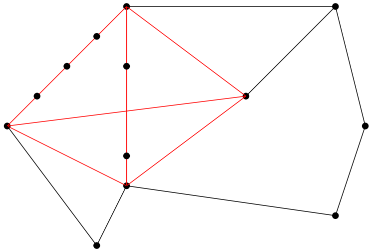
Definition

An *ISK4* in a graph G is an induced subdivision of K_4 in G. A graph is *ISK4-free* if it contains no induced subdivision of K_4.



Definition

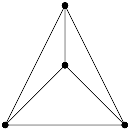
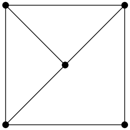
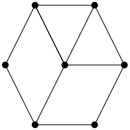
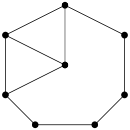
An *ISK4* in a graph G is an induced subdivision of K_4 in G. A graph is *ISK4-free* if it contains no induced subdivision of K_4.



Remark: An ISK4-free graph is in particular K_4-free, so it has no cliques of size greater than 3.

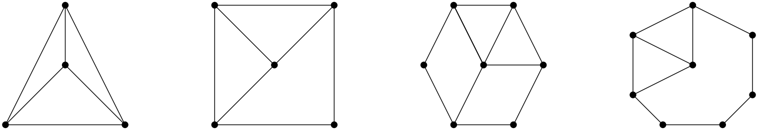
Definition

A *wheel* is a graph that consists of a chordless cycle and an additional vertex that has at least three neighbors in the cycle. A graph is *wheel-free* if it contains no wheel as an induced subgraph.

Definition

A *wheel* is a graph that consists of a chordless cycle and an additional vertex that has at least three neighbors in the cycle. A graph is *wheel-free* if it contains no wheel as an induced subgraph.



Definition

A graph is $\{ISK4, wheel\}$*-free* if it is both ISK4-free and wheel-free.

Theorem [Milanič, P., Trotignon, 2015+]

There is an algorithm with the following specifications:

- Input: A weighted {ISK4,wheel}-free graph (G, w) [a];
- Output: $\alpha(G, w)$ [b];
- Running time: $O(n^7)$, where $n = |V(G)|$.

[a] The *weight function* w assigns a non-negative integer weight $w(v)$ to each vertex v of G.

[b] $\alpha(G, w)$ is the maximum weight of a stable set (i.e. a set of pairwise non-adjacent vertices) of G with respect to w.

State of the art for **wheel-free** graphs:

1. recognition is NP-complete (Diot, Tavenas, Trotignon, 2014);
2. maximum stable set problem is NP-complete (easy).

State of the art for **wheel-free** graphs:

1. recognition is NP-complete (Diot, Tavenas, Trotignon, 2014);
2. maximum stable set problem is NP-complete (easy).

State of the art for **ISK4-free** graphs:

1. unknown complexity of the following problems: recognition, maximum stable set, coloring;
2. decomposition theorem (Lévêque, Maffray, Trotignon, 2012).

State of the art for **wheel-free** graphs:

1. recognition is NP-complete (Diot, Tavenas, Trotignon, 2014);
2. maximum stable set problem is NP-complete (easy).

State of the art for **ISK4-free** graphs:

1. unknown complexity of the following problems: recognition, maximum stable set, coloring;
2. decomposition theorem (Lévêque, Maffray, Trotignon, 2012).

State of the art for {**ISK4,wheel**}**-free** graphs:

1. decomposition theorem for {ISK4,wheel}-free graphs (Lévêque, Maffray, Trotignon, 2012);
2. polynomial-time recognition algorithm for {ISK4,wheel}-free graphs (Lévêque, Maffray, Trotignon, 2012);
3. {ISK4,wheel}-free graphs are 3-colorable + polynomial-time algorithm to 3-color them (Lévêque, Maffray, Trotignon, 2012).

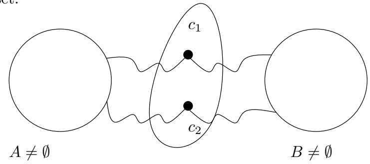
Theorem [Lévêque, Maffray, Trotignon, 2012]

If G is an {ISK4,wheel}-free graph, then either:

- G is a series-parallel graph[a], or
- G is the line graph of a chordless graph[b] of maximum degree at most three, or
- G is a complete bipartite graph, or
- G admits a clique-cutset, or
- G admits a proper 2-cutset.

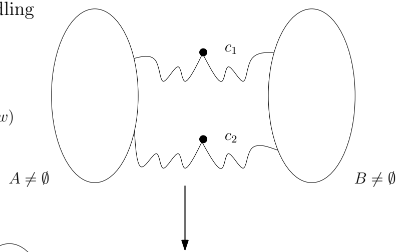
[a] series-parallel = no subdivision K_4 as a subgraph
[b] chordless = all cycles are induced

Proper 2-cutset:

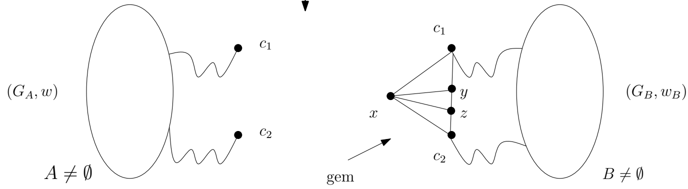


Neither $A \cup \{c_1, c_2\}$ nor $B \cup \{c_1, c_2\}$ induces a path between c_1 and c_2.

Attempt at handling proper 2-cutsets:



Need:
$\alpha(G_B, w_B) = \alpha(G, w)$



The weight of the gem gem vertices that lie inside a maximum weighted stable set of (G_B, w_B) is supposed to be equal to the weight of the part of a maximum weighted stable set of (G, w) that lies in (G_A, w).

$w_B(c_1) + w_B(c_2) = \alpha(G_A, w)$

$w_B(c_1) + w_B(z) = \alpha(G_A \setminus \{c_2\}, w)$

$w_B(c_2) + w_B(y) = \alpha(G_A \setminus \{c_1\}, w)$

$w_B(x) = \alpha(G_A \setminus \{c_1, c_2\}, w)$

$w_B(c_2) = w(c_2)$

Trigraphs to the rescue!

Trigraphs to the rescue!

- Trigraphs (introduced by Chudnovsky, 2003) are a certain generalization of graphs in which some pairs of vertices have "undetermined adjacency."

Trigraphs to the rescue!

- Trigraphs (introduced by Chudnovsky, 2003) are a certain generalization of graphs in which some pairs of vertices have "undetermined adjacency."
- There is a standard way to define $\{$ISK4,wheel$\}$-free trigraphs, and we proved a decomposition theorem for this class of trigraphs (similar to the graph case).

Trigraphs to the rescue!

- Trigraphs (introduced by Chudnovsky, 2003) are a certain generalization of graphs in which some pairs of vertices have "undetermined adjacency."
- There is a standard way to define {ISK4,wheel}-free trigraphs, and we proved a decomposition theorem for this class of trigraphs (similar to the graph case).
- We defined weighted trigraphs (we put weights on vertices and semi-adjacent pairs; motivated by proper 2-cutsets), and we constructed a polynomial-time algorithm that finds the maximum weight of a stable set in weighted {ISK4,wheel}-free trigraphs.

Trigraphs to the rescue!

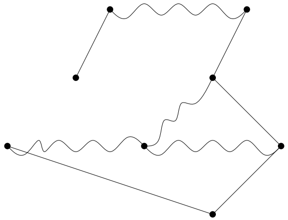
- Trigraphs (introduced by Chudnovsky, 2003) are a certain generalization of graphs in which some pairs of vertices have "undetermined adjacency."
- There is a standard way to define {ISK4,wheel}-free trigraphs, and we proved a decomposition theorem for this class of trigraphs (similar to the graph case).
- We defined weighted trigraphs (we put weights on vertices and semi-adjacent pairs; motivated by proper 2-cutsets), and we constructed a polynomial-time algorithm that finds the maximum weight of a stable set in weighted {ISK4,wheel}-free trigraphs.
    - Since every weighted {ISK4,wheel}-free graph is a weighted {ISK4,wheel}-free trigraph, this will yield a polynomial-time algorithm that finds the maximum weight of a stable set in a weighted {ISK4,wheel}-free graph.

Definition

A *trigraph* is a generalization of a graph in which there are three types of adjacency:

- strongly-adjacent pairs ("edges"),
- strongly anti-adjacent pairs ("non-edges"),
- semi-adjacent pairs ("optional edges" or "pairs of undetermined adjacency").

An *adjacent pair* is a pair or strongly-adjacent or semi-adjacent vertices. An *anti-adjacent pair* is a pair of strongly anti-adjacent or semi-adjacent vertices.

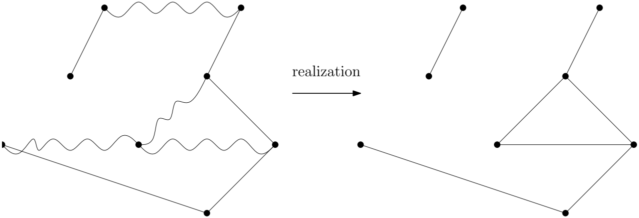


Remark: Every graph is a trigraph. (Indeed, a graph is simply a trigraph with no semi-adjacent pairs.)

Remark: Every graph is a trigraph. (Indeed, a graph is simply a trigraph with no semi-adjacent pairs.)

Definition

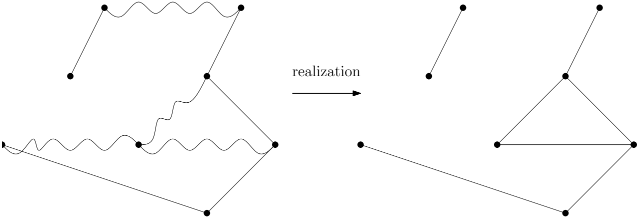
A *realization* of trigraph is any graph obtained by turning each semi-adjacent pair into an edge or a non-edge. So a trigraph with m semi-adjacent pairs has 2^m realizations.



Remark: Every graph is a trigraph. (Indeed, a graph is simply a trigraph with no semi-adjacent pairs.)

Definition

A *realization* of trigraph is any graph obtained by turning each semi-adjacent pair into an edge or a non-edge. So a trigraph with m semi-adjacent pairs has 2^m realizations.

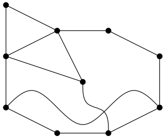


Definition

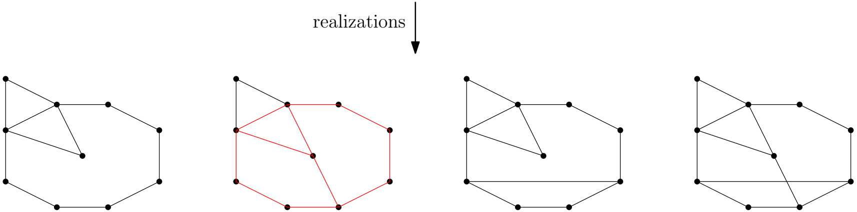
The *full realization* of trigraph is the graph obtained by turning all its semi-adjacent pairs into edges.

Definition

A trigraph is *ISK4-free* (resp. *wheel-free*, {*ISK4,wheel*}*-free*) if all its realizations are ISK4-free (resp. wheel-free, {ISK4,wheel}-free).

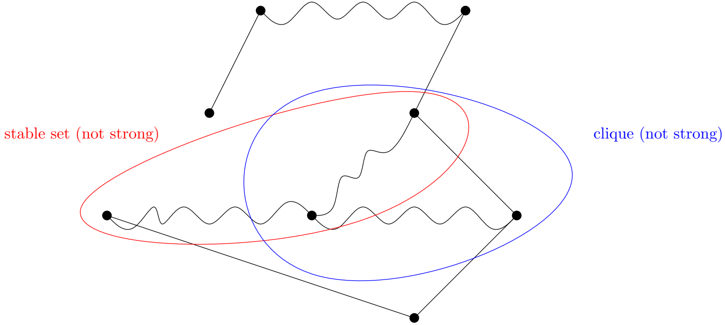


The trigraph is **not** wheel-free because it has a realization that is not wheel-free.



Definition

A *clique* in a trigraph is a set of pairwise adjacent (possibly semi-adjacent) vertices, and a *stable set* is a set of pairwise anti-adjacent (possibly semi-adjacent) vertices. A *strong clique* (resp. *strongly stable set*) is a clique (resp. stable set) with no semi-adjacent pairs.



We proved an "extreme decomposition theorem" that states that every $\{$ISK4,wheel$\}$-free trigraph is either "basic" or admits a "cutset" so that one of the "blocks of decomposition" is "basic."

We proved an “extreme decomposition theorem” that states that every $\{$ISK4,wheel$\}$-free trigraph is either “basic” or admits a “cutset” so that one of the “blocks of decomposition” is “basic.”

- Let’s define all this!

We proved an "extreme decomposition theorem" that states that every {ISK4,wheel}-free trigraph is either "basic" or admits a "cutset" so that one of the "blocks of decomposition" is "basic."

- Let's define all this!

Definition

A trigraph is *basic* if it is either

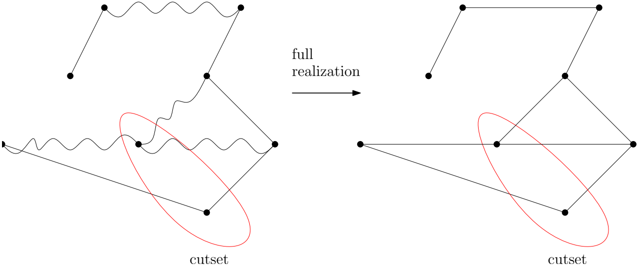
- a series-parallel trigraph (i.e. its full realization is a series-parallel graph), or
- a line trigraph[a] of a chordless graph of maximum degree at most three, or
- a complete bipartite graph.

---

[a] G is a *line trigraph* of a graph H if the full realization of G is the line graph of H, and no semi-adjacent pair of G is in a triangle.

Definition

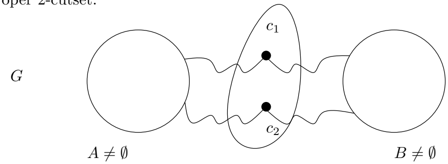
A trigraph is *connected* if its full realization is connected, and otherwise, it is *disconnected*. A *cutset* of a trigraph is a (possibly empty) set of vertices whose deletion yields a disconnected trigraph.



Theorem [Milanič, P., Trotignon, 2015+]

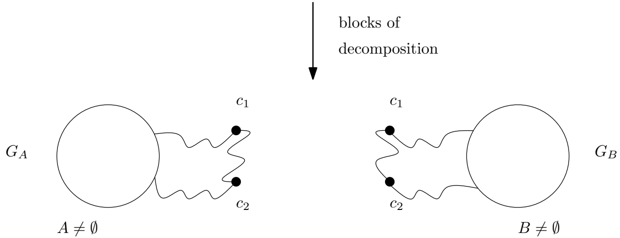
Every {ISK4,wheel}-free trigraph is either basic or admits a clique-cutset (i.e. a strong clique that is a cutset) or a proper 2-cutset s.t. one of the induced "blocks of decomposition" is basic.

Proper 2-cutset:



$c_1 c_2$ is an anti-adjacent pair (possibly semi-adjacent)

G is connected, and neither c_1 nor c_2 is a cut-vertex.



Fact: If G is {ISK4,wheel}-free, then so are G_A and G_B.

Theorem [Milanič, P., Trotignon, 2015+]

Every {ISK4,wheel}-free trigraph is either basic or admits a clique-cutset (i.e. a strong clique that is a cutset) or a proper 2-cutset s.t. one of the induced “blocks of decomposition” is basic.

Proof: Imitate the proof of the decomposition theorem for ISK4-free graphs (Lévêque, Maffray, Trotignon, 2012). Generalize to trigraphs, but(!) consider only the wheel-free case.

- The “jump” to trigraphs doesn’t complicate the proof much; the restriction to the wheel-free case significantly simplifies it.

A bit of extra work to get the “extreme” decomposition theorem.

- This is algorithmic! There is a polynomial-time algorithm that, given an {ISK4,wheel}-free trigraph G, either determines that G is basic, or finds an “extreme decomposition” of G via a clique-cutset or a proper 2-cutset. Q.E.D.

Onward to weighted trigraphs!

Onward to weighted trigraphs!

- Idea: We assign weights to vertices and to semi-adjacent pairs (each vertex gets one weight, and each semi-adjacent pair gets three weights).

Onward to weighted trigraphs!

- Idea: We assign weights to vertices and to semi-adjacent pairs (each vertex gets one weight, and each semi-adjacent pair gets three weights).
- Warning: The weight of a set S of vertices depends on the weights of the vertices inside S, and on the weights of all semi-adjacent pairs in the trigraph (and not just those inside the set S). This is needed because of proper 2-cutsets.

Onward to weighted trigraphs!

- Idea: We assign weights to vertices and to semi-adjacent pairs (each vertex gets one weight, and each semi-adjacent pair gets three weights).
- Warning: The weight of a set S of vertices depends on the weights of the vertices inside S, and on the weights of all semi-adjacent pairs in the trigraph (and not just those inside the set S). This is needed because of proper 2-cutsets.
- However: In the case of graphs (i.e. trigraphs with no semi-adjacent pairs), we assign weights to vertices only, and so we get an ordinary weighted graph (and the weight of a set is calculated in the usual way: by summing up the weights of vertices inside the set).

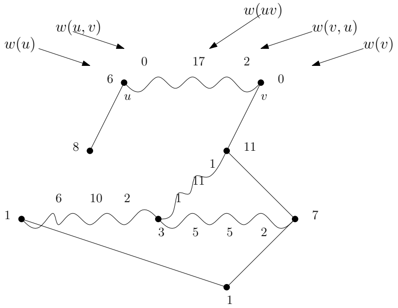
Onward to weighted trigraphs!

- Idea: We assign weights to vertices and to semi-adjacent pairs (each vertex gets one weight, and each semi-adjacent pair gets three weights).
- Warning: The weight of a set S of vertices depends on the weights of the vertices inside S, and on the weights of all semi-adjacent pairs in the trigraph (and not just those inside the set S). This is needed because of proper 2-cutsets.
- However: In the case of graphs (i.e. trigraphs with no semi-adjacent pairs), we assign weights to vertices only, and so we get an ordinary weighted graph (and the weight of a set is calculated in the usual way: by summing up the weights of vertices inside the set).
  - Thus, we can plug in a weighted {ISK4,wheel}-free graph (G, w) into our algorithm for trigraphs and get an "ordinary" $\alpha(G, w)$ for weighted graphs.

Definition

A *weighted trigraph* is an ordered pair (G, w) s.t. G is a trigraph, and w is a *weight function* for G s.t.

- to each vertex v of G, w assigns a non-negative integer weight $w(v)$, and
- to each semi-adjacent pair uv of G, w assigns three non-negative integer weights, $w(uv)$, $w(u, v)$, and $w(v, u)$, and these weights satisfy $w(u, v), w(v, u) \leq w(uv)$.

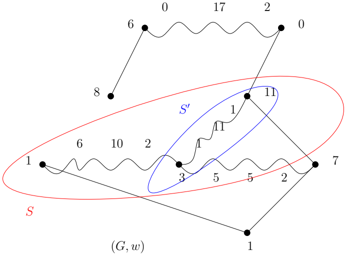


Definition

The *weight* of a set S of vertices in a weighted trigraph (G, w) is the sum of the following three quantities:

- the sum of all $w(u)$ s.t. $u \in S$;
- the sum of all $w(u, v)$ s.t. uv is a semi-adjacent pair of G with $u \in S$ and $v \notin S$;
- the sum of all $w(uv)$ s.t. uv is a semi-adjacent pair of G with $u, v \notin S$.

$\alpha(G, w)$ is the maximum weight of a stable set of (G, w).



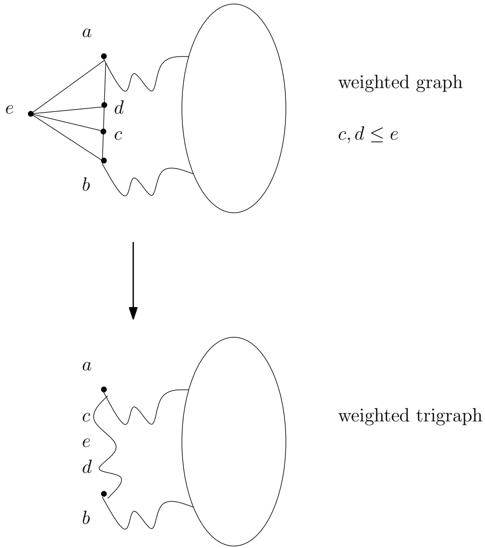
weight of S in (G, w):

$(1{+}3{+}11){+}5{+}17 = 37$

weight of S' in (G, w):

$(3 + 11) + (2 + 5) + 17 = 38$

Semi-adjacent pairs can imitate gems! (But without increasing the number of vertices, and without introducing wheels.)



Algorithm:

- Input: A weighted $\{$ISK4,wheel$\}$-free trigraph (G, w);
- Output: $\alpha(G, w)$;
- Running time: $O(n^7)$, where $n = |V(G)|$.

Algorithm:

- Input: A weighted $\{\text{ISK4,wheel}\}$-free trigraph (G, w);
- Output: $\alpha(G, w)$;
- Running time: $O(n^7)$, where $n = |V(G)|$.

Outline:

Algorithm:

- Input: A weighted $\{$ISK4,wheel$\}$-free trigraph (G, w);
- Output: $\alpha(G, w)$;
- Running time: $O(n^7)$, where $n = |V(G)|$.

Outline:

1. Check if G is basic, and if so, find $\alpha(G, w)$ directly, and stop.
    - This involves transforming the weighted trigraph (G, w) into a weighted graph that has the same α, and then finding α in that graph.

Algorithm:

- Input: A weighted {ISK4,wheel}-free trigraph (G, w);
- Output: $\alpha(G, w)$;
- Running time: $O(n^7)$, where $n = |V(G)|$.

Outline:

1. Check if G is basic, and if so, find $\alpha(G, w)$ directly, and stop.
   - This involves transforming the weighted trigraph (G, w) into a weighted graph that has the same α, and then finding α in that graph.
2. If not, then find an extreme decomposition (via a clique-cutset or a proper 2-cutset).

Algorithm:

- Input: A weighted {ISK4,wheel}-free trigraph (G, w);
- Output: $\alpha(G, w)$;
- Running time: $O(n^7)$, where $n = |V(G)|$.

Outline:

1. Check if G is basic, and if so, find $\alpha(G, w)$ directly, and stop.
    - This involves transforming the weighted trigraph (G, w) into a weighted graph that has the same α, and then finding α in that graph.
2. If not, then find an extreme decomposition (via a clique-cutset or a proper 2-cutset).
3. Compute α in the basic block and some of its induced subtrigraphs (possibly with slightly modified weights).

Algorithm:

- Input: A weighted $\{$ISK4,wheel$\}$-free trigraph (G, w);
- Output: $\alpha(G, w)$;
- Running time: $O(n^7)$, where $n = |V(G)|$.

Outline:

1. Check if G is basic, and if so, find $\alpha(G, w)$ directly, and stop.
    - This involves transforming the weighted trigraph (G, w) into a weighted graph that has the same α, and then finding α in that graph.
2. If not, then find an extreme decomposition (via a clique-cutset or a proper 2-cutset).
3. Compute α in the basic block and some of its induced subtrigraphs (possibly with slightly modified weights).
4. Then change weights in the other block, and (recursively) compute α.

That's all.

Thanks for listening!