

# Decomposing Star-like Cubic Graphs

Oliver Bachtler, Sven O. Krumke

RPTU Kaiserslautern-Landau

27. SEG Workshop, 2023

# Outline

## Basics

- 3-decompositions

- Existence for Hamiltonian graphs

## Constructing a Decomposition

- From the centre

- To the tips

# 3-Decompositions of Graphs

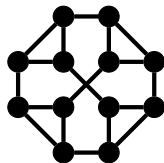
## Definition (3-Decomposition)

A **3-decomposition** of a connected cubic graph  $G$  consists of

# 3-Decompositions of Graphs

## Definition (Cubic Graphs)

A graph is **cubic** if every vertex has degree 3.



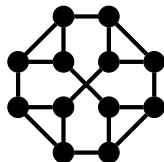
## Definition (3-Decomposition)

A **3-decomposition** of a connected cubic graph  $G$  consists of

# 3-Decompositions of Graphs

## Definition (Cubic Graphs)

A graph is **cubic** if every vertex has degree 3.



## Definition (3-Decomposition)

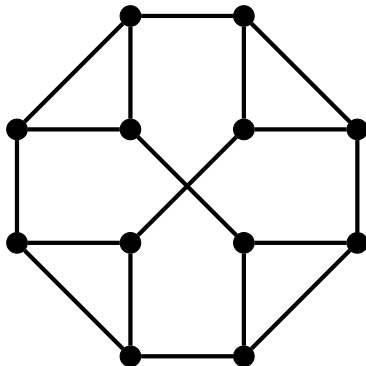
A **3-decomposition** of a connected cubic graph  $G$  consists of

- ▶ a spanning tree  $T$ ,
- ▶ a set of cycles  $C$ , and
- ▶ a matching  $M$

such that  $E(G)$  is the disjoint union  $E(T) \cup E(C) \cup M$ .

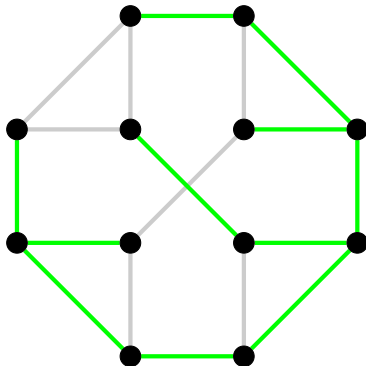
# An Example Graph

- ▶ Given: connected cubic graph.



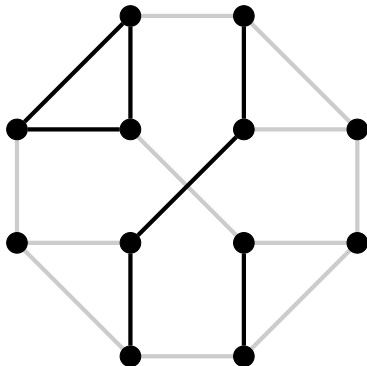
# An Example Graph

- ▶ Given: connected cubic graph.
- ▶ Take a **spanning tree**.



# An Example Graph

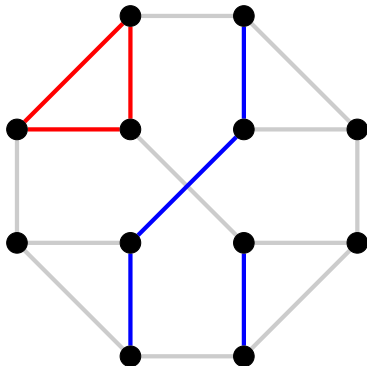
- ▶ Given: connected cubic graph.
- ▶ Take a **spanning tree**.





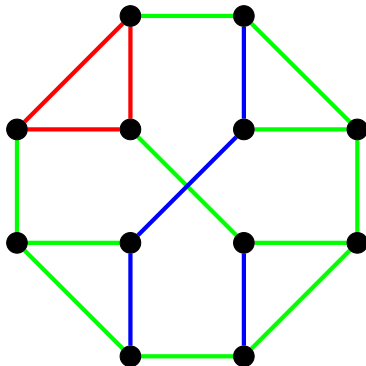
# An Example Graph

- ▶ Given: connected cubic graph.
- ▶ Take a **spanning tree**.
- ▶ The remaining edges form **cycles** and **paths**.



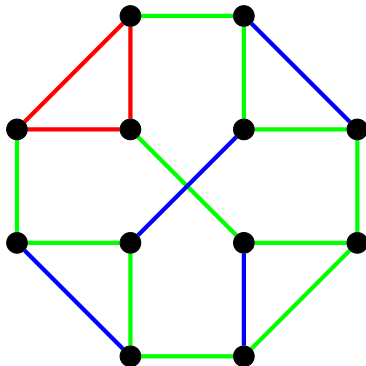
# An Example Graph

- ▶ Given: connected cubic graph.
- ▶ Take a **spanning tree**.
- ▶ The remaining edges form **cycles** and **paths**.
- ▶ Want paths of length 1.



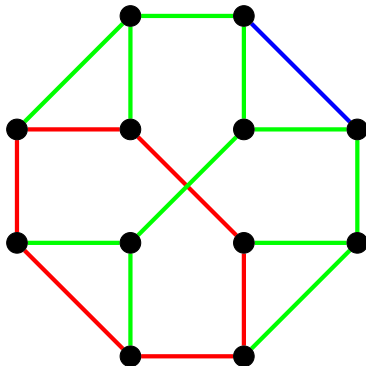
# An Example Graph

- ▶ Given: connected cubic graph.
- ▶ Take a **spanning tree**.
- ▶ The remaining edges form **cycles** and **paths**.
- ▶ Want paths of length 1.



# An Example Graph

- ▶ Given: connected cubic graph.
- ▶ Take a **spanning tree**.
- ▶ The remaining edges form **cycles** and **paths**.
- ▶ Want paths of length 1.



# The 3-Decomposition Conjecture

## Definition (3-Decomposition)

A **3-decomposition** of a connected cubic graph  $G$  consists of

- ▶ a spanning tree  $T$ ,
- ▶ a set of cycles  $C$ , and
- ▶ a matching  $M$

such that  $E(G)$  is the disjoint union  $E(T) \cup E(C) \cup M$ .

# The 3-Decomposition Conjecture

## Definition (3-Decomposition)

A **3-decomposition** of a connected cubic graph  $G$  consists of

- ▶ a spanning tree  $T$ ,
- ▶ a set of cycles  $C$ , and
- ▶ a matching  $M$

such that  $E(G)$  is the disjoint union  $E(T) \cup E(C) \cup M$ .

## Conjecture (3-Decomposition Conjecture, 2011)

*Every connected cubic graph has a 3-decomposition.*

# The 3-Decomposition Conjecture

## Definition (3-Decomposition)

A **3-decomposition** of a connected cubic graph  $G$  consists of

- ▶ a spanning tree  $T$ ,
- ▶ a set of cycles  $C$ , and
- ▶ a matching  $M$

such that  $E(G)$  is the disjoint union  $E(T) \cup E(C) \cup M$ .

## Conjecture (3-Decomposition Conjecture, 2011)

*Every connected cubic graph has a 3-decomposition.*

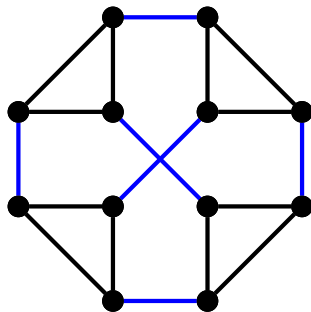
## Theorem (Akbari, Jensen, Siggers, 2015)

*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*

# Our Result

## Definition (Star-like Graphs)

Let  $G$  be a connected cubic graph with a perfect matching  $M$ .



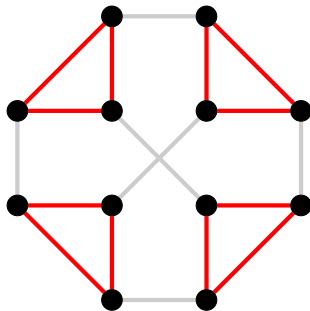


# Our Result

## Definition (Star-like Graphs)

Let  $G$  be a connected cubic graph with a perfect matching  $M$ .

- ▶  $G - M$  consists of cycles.

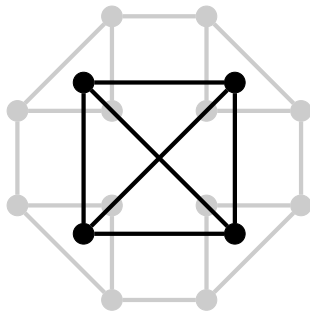


# Our Result

## Definition (Star-like Graphs)

Let  $G$  be a connected cubic graph with a perfect matching  $M$ .

- ▶  $G - M$  consists of cycles.
- ▶ Contracting these yields the **contraction graph**  $G_M$ .



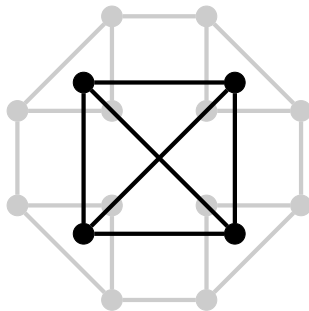
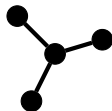
# Our Result

## Definition (Star-like Graphs)

Let  $G$  be a connected cubic graph with a perfect matching  $M$ .

- ▶  $G - M$  consists of cycles.
- ▶ Contracting these yields the contraction graph  $G_M$ .

$G$  is **star-like** if it has a perfect matching  $M$  such that  $G_M$  is a star.



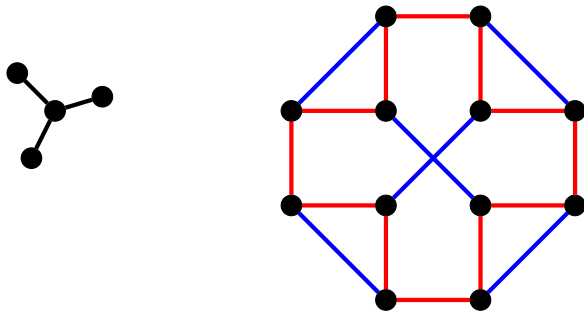
# Our Result

## Definition (Star-like Graphs)

Let  $G$  be a connected cubic graph with a perfect matching  $M$ .

- ▶  $G - M$  consists of cycles.
- ▶ Contracting these yields the **contraction graph**  $G_M$ .

$G$  is **star-like** if it has a perfect matching  $M$  such that  $G_M$  is a star.



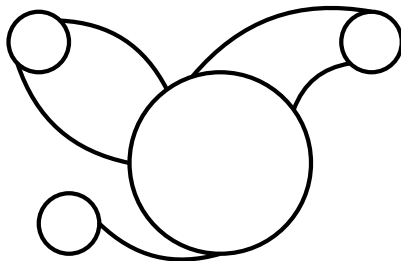
# Our Result

## Definition (Star-like Graphs)

Let  $G$  be a connected cubic graph with a perfect matching  $M$ .

- ▶  $G - M$  consists of cycles.
- ▶ Contracting these yields the contraction graph  $G_M$ .

$G$  is **star-like** if it has a perfect matching  $M$  such that  $G_M$  is a star.



# Our Result

## Definition (Star-like Graphs)

Let  $G$  be a connected cubic graph with a perfect matching  $M$ .

- ▶  $G - M$  consists of cycles.
- ▶ Contracting these yields the contraction graph  $G_M$ .

$G$  is **star-like** if it has a perfect matching  $M$  such that  $G_M$  is a star.

## Definition (3-Connectivity)

A graph  $G$  is **3-connected** if removing 2 vertices does not disconnect it.

# Our Result

## Definition (Star-like Graphs)

Let  $G$  be a connected cubic graph with a perfect matching  $M$ .

- ▶  $G - M$  consists of cycles.
- ▶ Contracting these yields the contraction graph  $G_M$ .

$G$  is **star-like** if it has a perfect matching  $M$  such that  $G_M$  is a star.

## Definition (3-Connectivity)

A graph  $G$  is **3-connected** if removing 2 vertices does not disconnect it.

## Theorem (Xie, Zhou, Zhou, 2020)

*Every graph with a contraction graph on 3 vertices has a 3-decomposition.*

# Our Result

## Definition (Star-like Graphs)

Let  $G$  be a connected cubic graph with a perfect matching  $M$ .

- ▶  $G - M$  consists of cycles.
- ▶ Contracting these yields the contraction graph  $G_M$ .

$G$  is **star-like** if it has a perfect matching  $M$  such that  $G_M$  is a star.

## Definition (3-Connectivity)

A graph  $G$  is **3-connected** if removing 2 vertices does not disconnect it.

## Theorem (Main Result, 2022)

*Every 3-connected star-like graph has a 3-decomposition.*



# Hamiltonian Graphs have 3-Decompositions

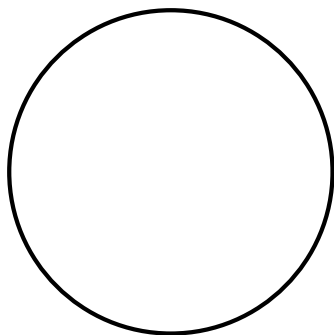
Theorem (Akbari, Jensen, Siggers)

*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*

# Hamiltonian Graphs have 3-Decompositions

Theorem (Akbari, Jensen, Siggers)

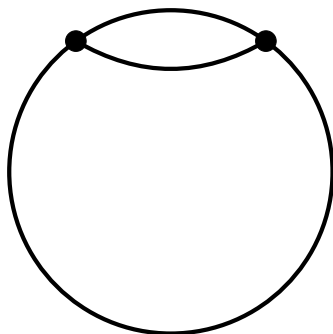
*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*



# Hamiltonian Graphs have 3-Decompositions

Theorem (Akbari, Jensen, Siggers)

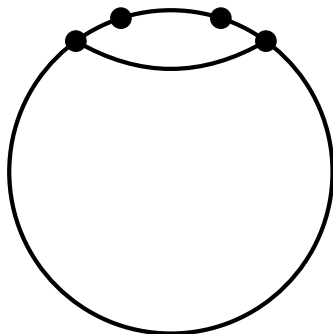
*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*



# Hamiltonian Graphs have 3-Decompositions

Theorem (Akbari, Jensen, Siggers)

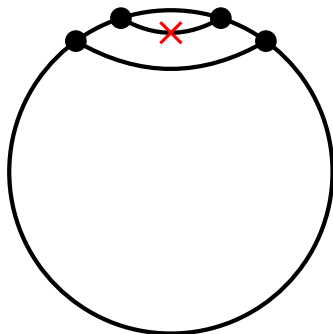
*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*



# Hamiltonian Graphs have 3-Decompositions

Theorem (Akbari, Jensen, Siggers)

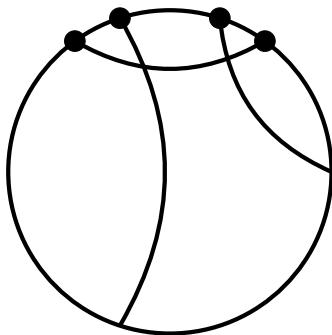
*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*



# Hamiltonian Graphs have 3-Decompositions

Theorem (Akbari, Jensen, Siggers)

*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*

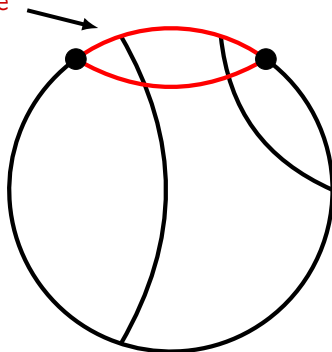


# Hamiltonian Graphs have 3-Decompositions

Theorem (Akbari, Jensen, Siggers)

*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*

minimal cycle

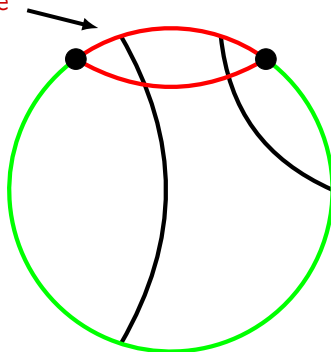


# Hamiltonian Graphs have 3-Decompositions

Theorem (Akbari, Jensen, Siggers)

*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*

minimal cycle



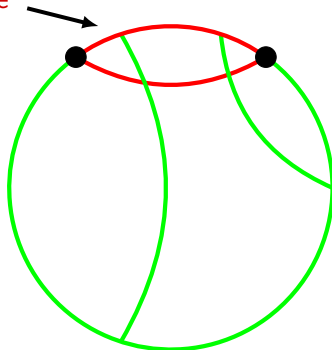


# Hamiltonian Graphs have 3-Decompositions

Theorem (Akbari, Jensen, Siggers)

*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*

minimal cycle

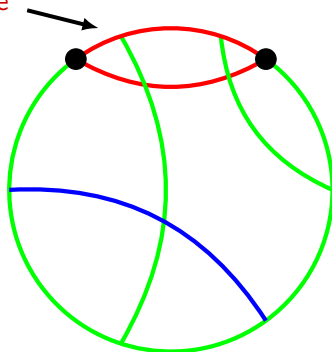


# Hamiltonian Graphs have 3-Decompositions

Theorem (Akbari, Jensen, Siggers)

*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*

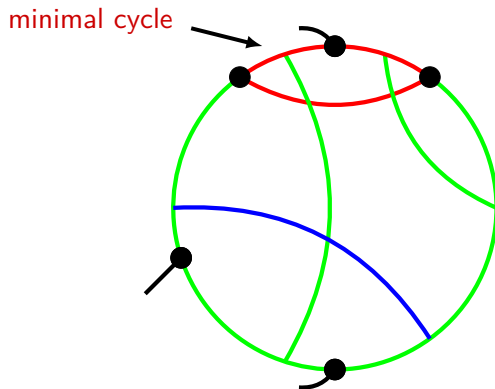
minimal cycle



# Hamiltonian Graphs have 3-Decompositions

Theorem (Akbari, Jensen, Siggers)

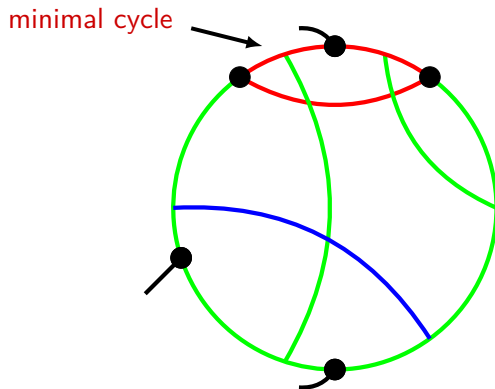
*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*



# Hamiltonian Graphs have 3-Decompositions

Theorem (Akbari, Jensen, Siggers)

*Every connected cubic **Hamiltonian** graph has a 3-decomposition.*



Call this the **decomposition given by the minimal cycle**.

# First Steps

## Theorem (Main Result)

*Every 3-connected star-like graph has a 3-decomposition.*

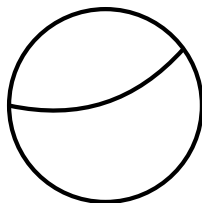
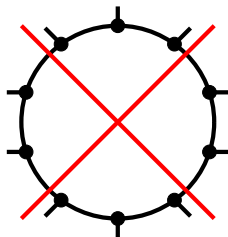
# First Steps

## Theorem (Main Result)

Every *3-connected star-like* graph has a 3-decomposition.

## Assumption

All cycles in  $G - M$  have chords in  $G$ .



# First Steps

## Theorem (Main Result)

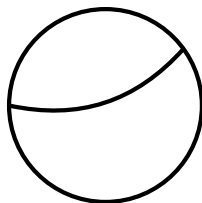
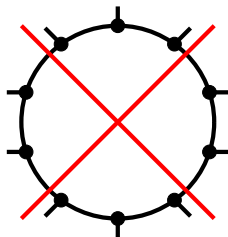
Every *3-connected star-like* graph has a 3-decomposition.

## Assumption

All cycles in  $G - M$  have chords in  $G$ .

## Observation

There exists a minimal cycle avoiding *any* fixed boundary vertex.



# First Steps

## Theorem (Main Result)

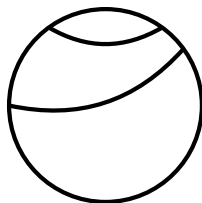
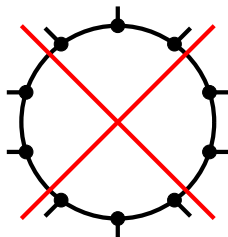
Every *3-connected star-like* graph has a 3-decomposition.

## Assumption

All cycles in  $G - M$  have chords in  $G$ .

## Observation

There exists a minimal cycle avoiding *any* fixed boundary vertex.





# First Steps

## Theorem (Main Result)

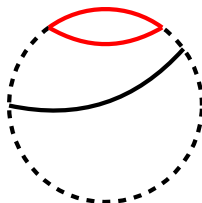
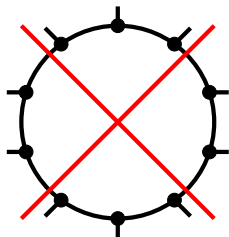
Every *3-connected star-like* graph has a 3-decomposition.

## Assumption

All cycles in  $G - M$  have chords in  $G$ .

## Observation

There exists a minimal cycle avoiding *any* fixed boundary vertex.



# First Steps

## Theorem (Main Result)

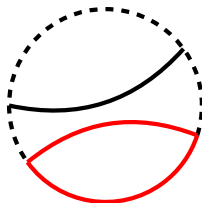
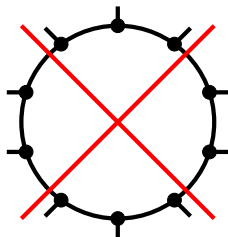
Every *3-connected star-like* graph has a 3-decomposition.

## Assumption

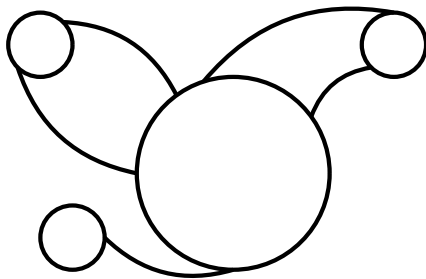
All cycles in  $G - M$  have chords in  $G$ .

## Observation

There exists a minimal cycle avoiding *any* fixed boundary vertex.

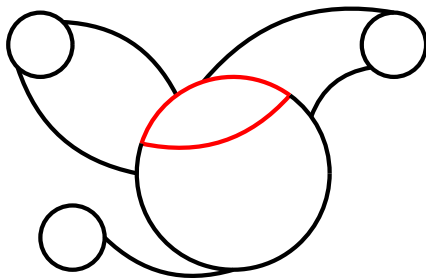


# Decomposing the Centre



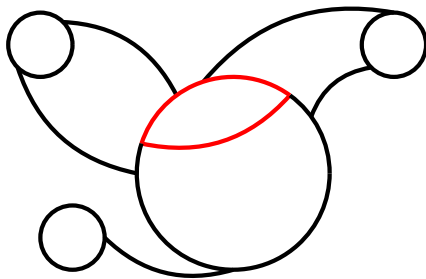
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .



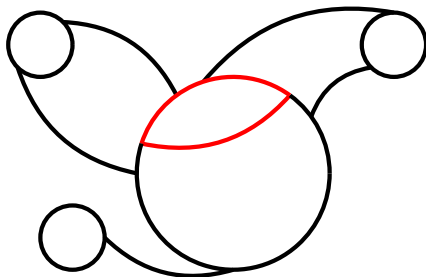
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.



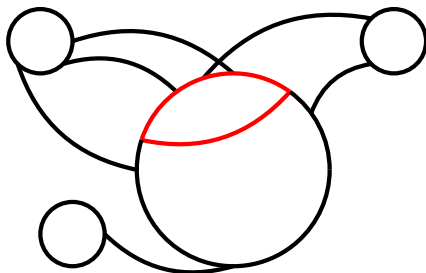
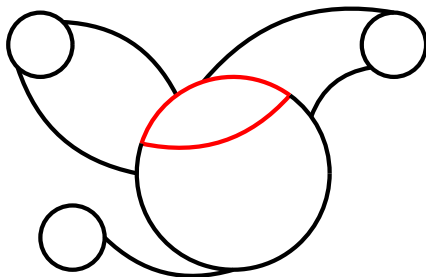
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.



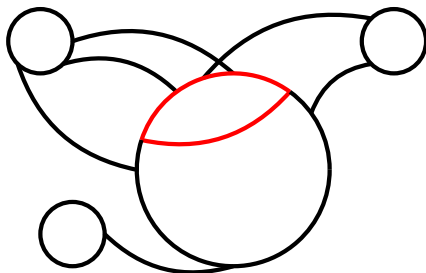
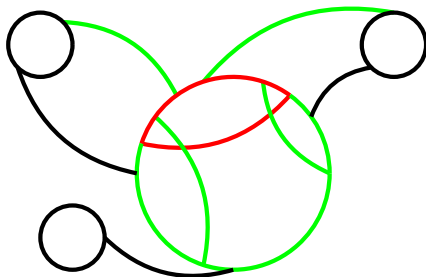
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.



# Decomposing the Centre

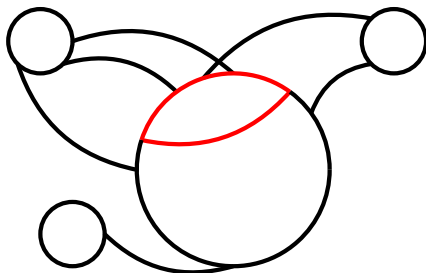
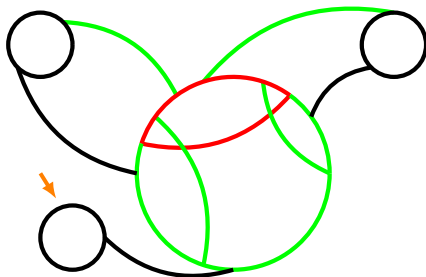
- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .





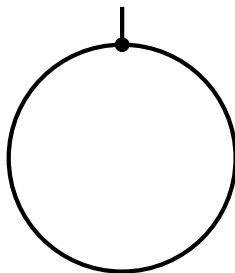
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,



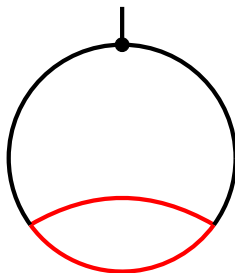
## The Tips: Only Black Edges

Take the decomposition given by a minimal cycle:



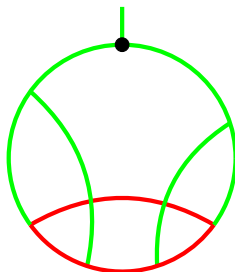
# The Tips: Only Black Edges

Take the decomposition given by a minimal cycle:



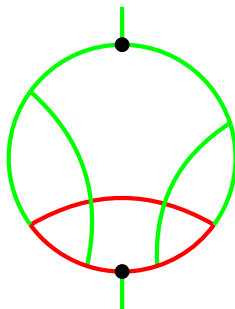
# The Tips: Only Black Edges

Take the decomposition given by a minimal cycle:



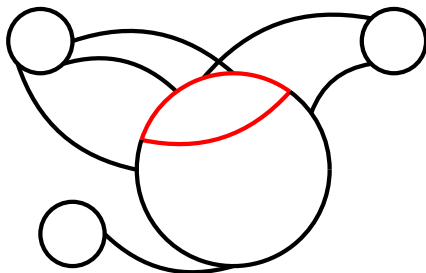
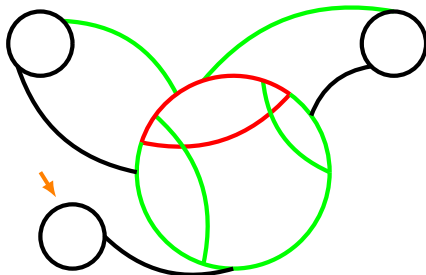
# The Tips: Only Black Edges

Take the decomposition given by a minimal cycle:



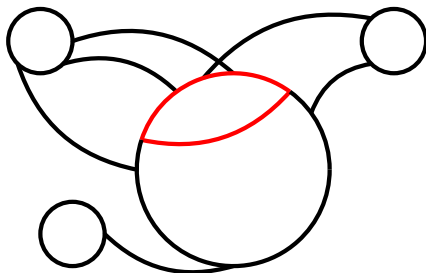
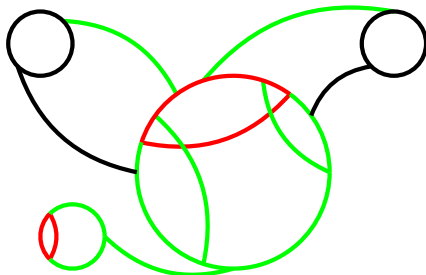
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,



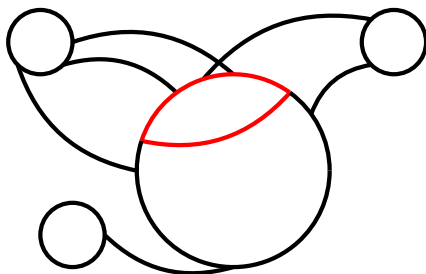
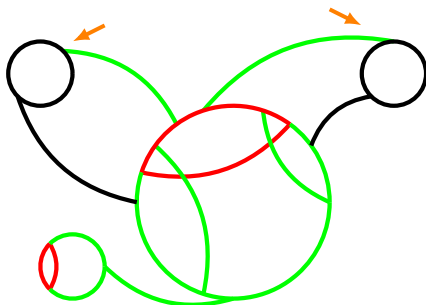
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,



# Decomposing the Centre

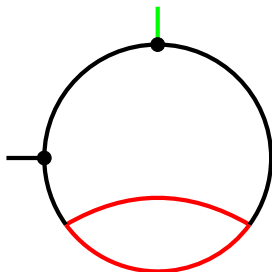
- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,
  - ▶ 1 green edge,





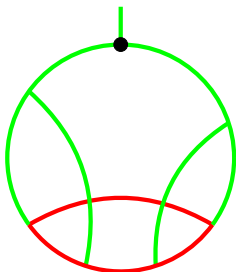
# The Tips: One Green Edge

- ▶ Look for a useful minimal cycle.



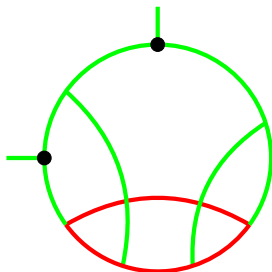
# The Tips: One Green Edge

- Look for a useful minimal cycle.



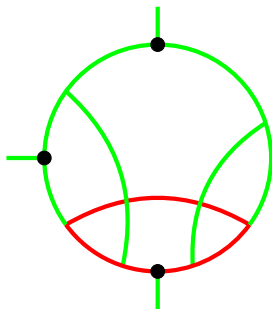
# The Tips: One Green Edge

- Look for a useful minimal cycle.



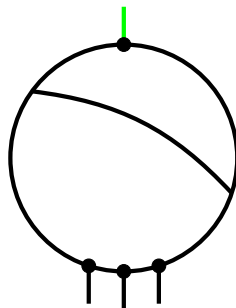
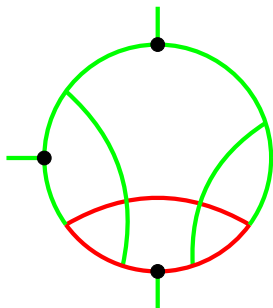
# The Tips: One Green Edge

- Look for a useful minimal cycle.



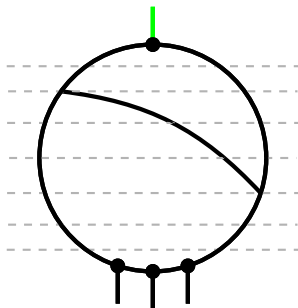
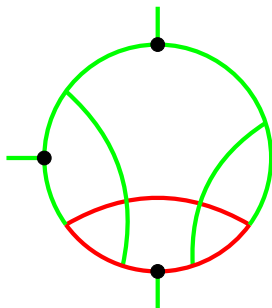
# The Tips: One Green Edge

- ▶ Look for a useful minimal cycle.
- ▶ If none exists  $\Rightarrow$  All chords go from the left to the right path.



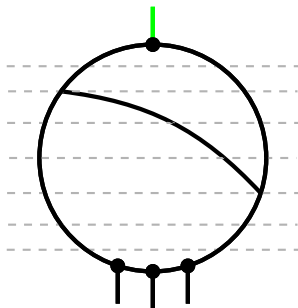
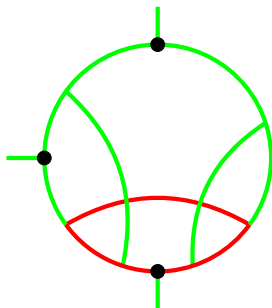
# The Tips: One Green Edge

- ▶ Look for a useful minimal cycle.
- ▶ If none exists  $\Rightarrow$  All chords go from the left to the right path.
- ▶ Both paths have the same length  $\Rightarrow$  Levels.



# The Tips: One Green Edge

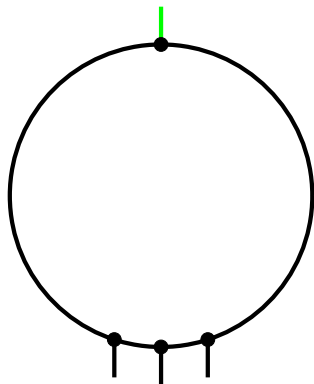
- ▶ Look for a useful minimal cycle.
- ▶ If none exists  $\Rightarrow$  All chords go from the left to the right path.
- ▶ Both paths have the same length  $\Rightarrow$  Levels.
- ▶ Call a chord **long** if its ends are at least 2 levels apart.



# The Tips: One Green Edge

Case 1: No long chord exists

- ▶ Two form of chords:
  - ▶ direct
  - ▶ cross



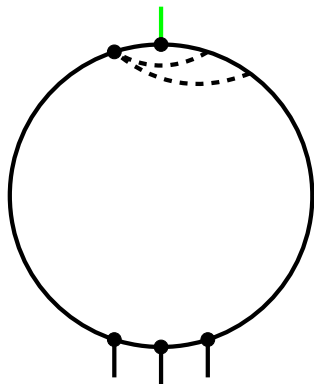


# The Tips: One Green Edge

Case 1: No long chord exists

► Two form of chords:

- direct
- cross

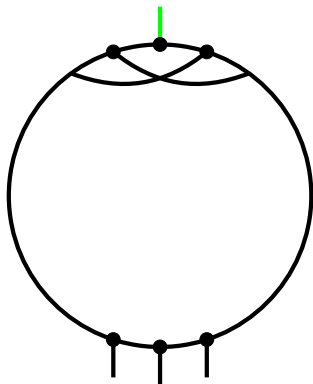


# The Tips: One Green Edge

Case 1: No long chord exists

► Two form of chords:

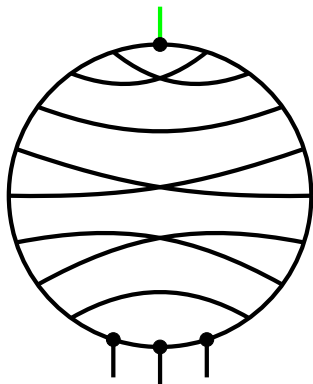
- direct
- cross



# The Tips: One Green Edge

Case 1: No long chord exists

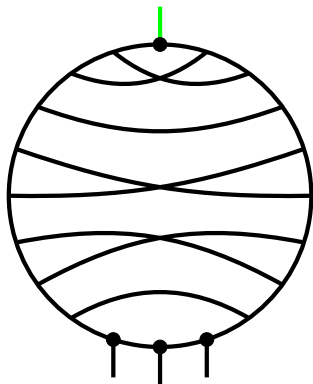
- ▶ Two form of chords:
  - ▶ direct
  - ▶ cross



# The Tips: One Green Edge

Case 1: No long chord exists

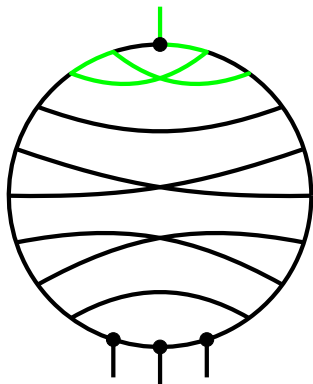
- ▶ Two form of chords:
  - ▶ direct
  - ▶ cross
- ▶ Obtain a Hamiltonian path.



# The Tips: One Green Edge

Case 1: No long chord exists

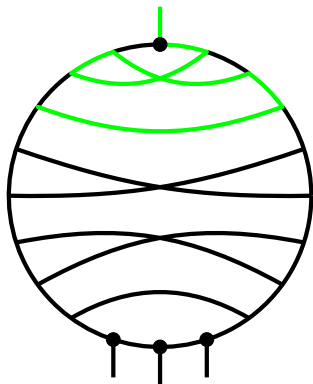
- ▶ Two form of chords:
  - ▶ direct
  - ▶ cross
- ▶ Obtain a Hamiltonian path.



# The Tips: One Green Edge

Case 1: No long chord exists

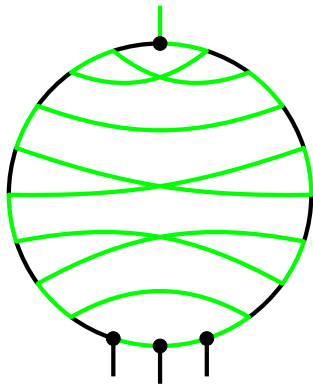
- ▶ Two form of chords:
  - ▶ direct
  - ▶ cross
- ▶ Obtain a Hamiltonian path.



# The Tips: One Green Edge

Case 1: No long chord exists

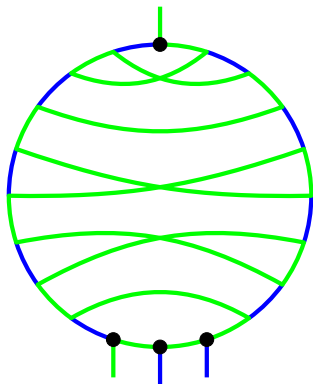
- ▶ Two form of chords:
  - ▶ direct
  - ▶ cross
- ▶ Obtain a Hamiltonian path.



# The Tips: One Green Edge

Case 1: No long chord exists

- ▶ Two form of chords:
  - ▶ direct
  - ▶ cross
- ▶ Obtain a Hamiltonian path.

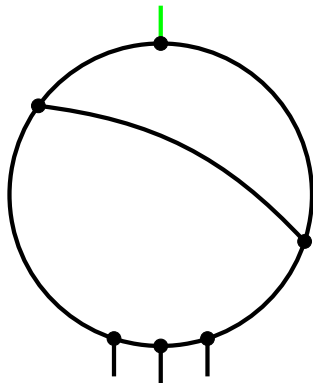




# The Tips: One Green Edge

Case 2: A long chord exists

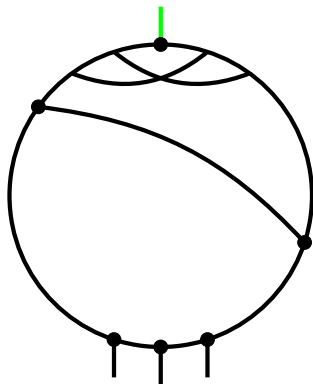
- Take the **first** such chord.



# The Tips: One Green Edge

Case 2: A long chord exists

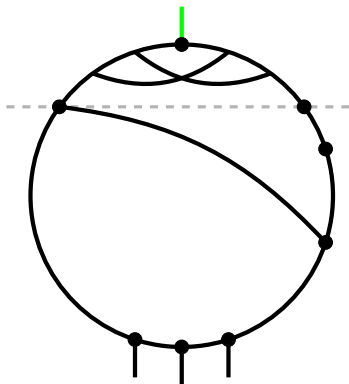
- ▶ Take the **first** such chord.
- ▶ Vertices before paired up.



# The Tips: One Green Edge

Case 2: A long chord exists

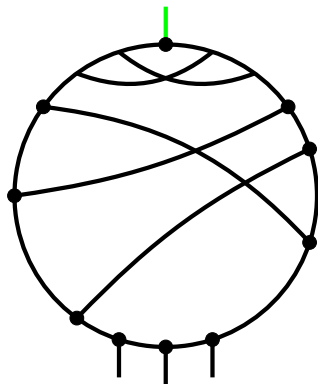
- ▶ Take the **first** such chord.
- ▶ Vertices before paired up.
- ▶ Regard the other vertex on this level and its successor.



# The Tips: One Green Edge

Case 2: A long chord exists

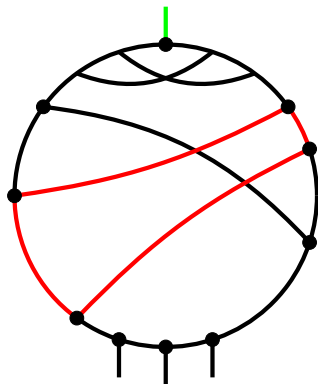
- ▶ Take the **first** such chord.
- ▶ Vertices before paired up.
- ▶ Regard the other vertex on this level and its successor.
- ▶ Neighbours below the chord.



# The Tips: One Green Edge

Case 2: A long chord exists

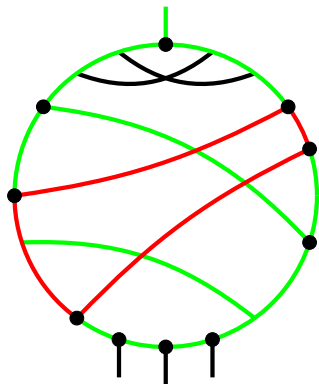
- ▶ Take the **first** such chord.
- ▶ Vertices before paired up.
- ▶ Regard the other vertex on this level and its successor.
- ▶ Neighbours below the chord.
- ▶ Use a two-chord cycle.



# The Tips: One Green Edge

## Case 2: A long chord exists

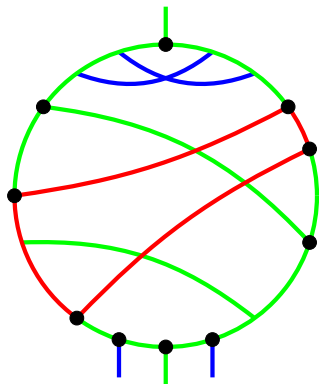
- ▶ Take the **first** such chord.
- ▶ Vertices before paired up.
- ▶ Regard the other vertex on this level and its successor.
- ▶ Neighbours below the chord.
- ▶ Use a two-chord cycle.
- ▶ Leaves a tree.



# The Tips: One Green Edge

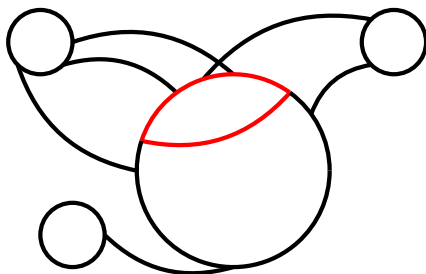
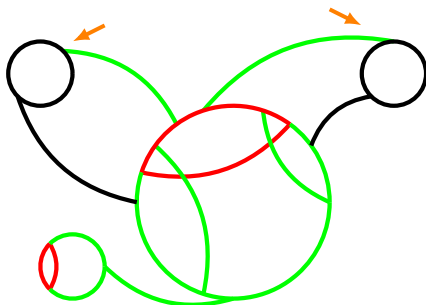
## Case 2: A long chord exists

- ▶ Take the **first** such chord.
- ▶ Vertices before paired up.
- ▶ Regard the other vertex on this level and its successor.
- ▶ Neighbours below the chord.
- ▶ Use a two-chord cycle.
- ▶ Leaves a tree.



# Decomposing the Centre

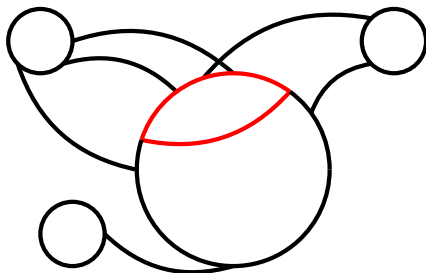
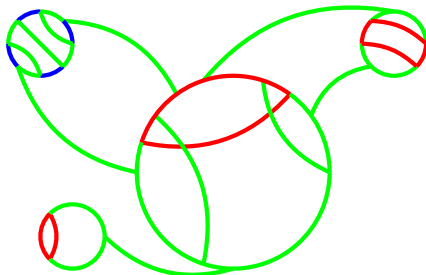
- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,
  - ▶ 1 green edge,





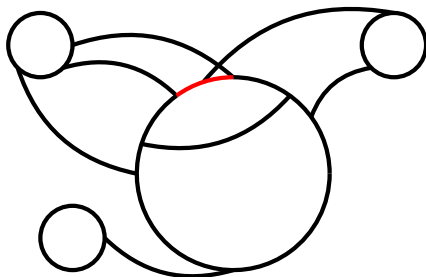
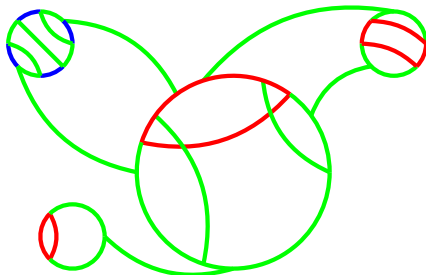
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,
  - ▶ 1 green edge,



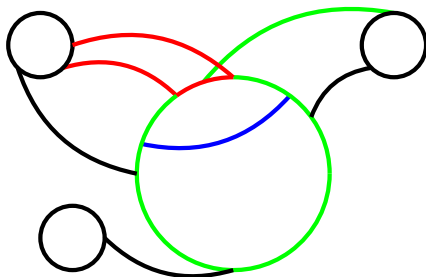
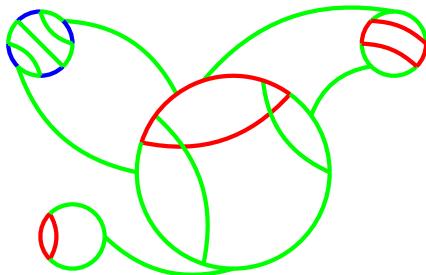
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
  - ▶ using a part of  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,
  - ▶ 1 green edge,



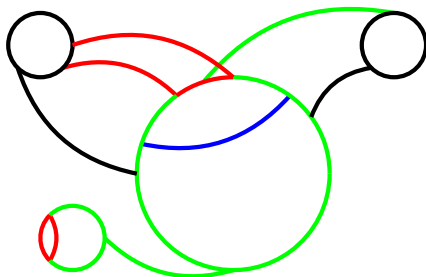
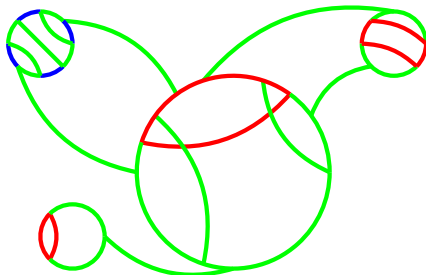
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
  - ▶ using a part of  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,
  - ▶ 1 green edge,



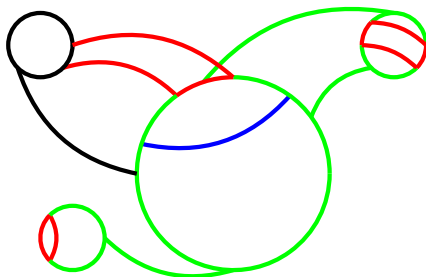
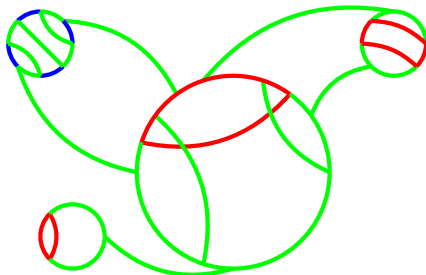
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
  - ▶ using a part of  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,
  - ▶ 1 green edge,



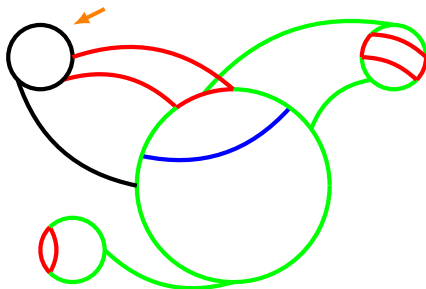
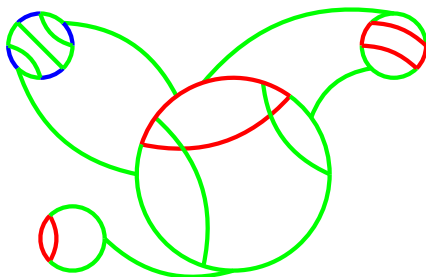
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
  - ▶ using a part of  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,
  - ▶ 1 green edge,

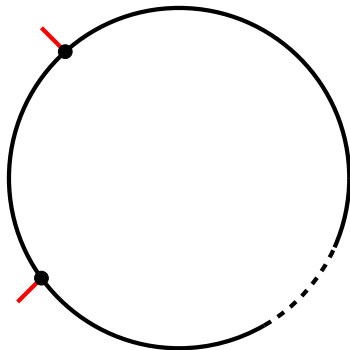


# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
  - ▶ using a part of  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,
  - ▶ 1 green edge,
  - ▶ 2 red edges.

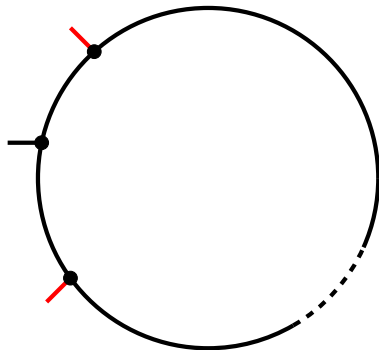


# The Tips: Two Red Edges



# The Tips: Two Red Edges

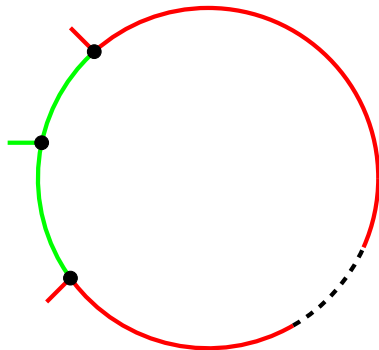
- ▶ One path connects to centre.





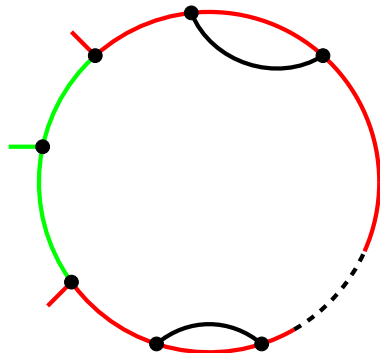
## The Tips: Two Red Edges

- ▶ One path connects to centre.
- ▶ Put it in  $T$  and rest in  $C$ .



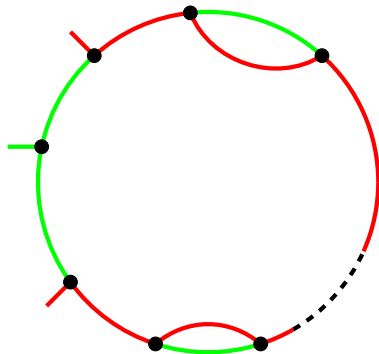
# The Tips: Two Red Edges

- ▶ One path connects to centre.
- ▶ Put it in  $\mathcal{T}$  and rest in  $\mathcal{C}$ .
- ▶ Problem: “red” chords.



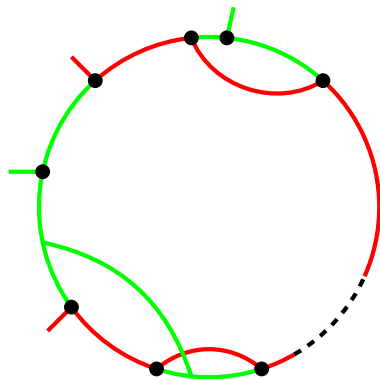
# The Tips: Two Red Edges

- ▶ One path connects to centre.
- ▶ Put it in  $\mathcal{T}$  and rest in  $\mathcal{C}$ .
- ▶ Problem: “red” chords.
- ▶ Idea: use to shortcut.



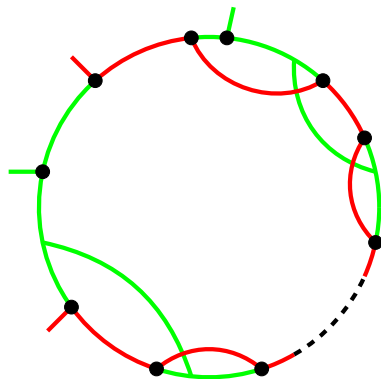
# The Tips: Two Red Edges

- ▶ One path connects to centre.
- ▶ Put it in  $T$  and rest in  $C$ .
- ▶ Problem: “red” chords.
- ▶ Idea: use to shortcut.
- ▶ Must connect new paths.



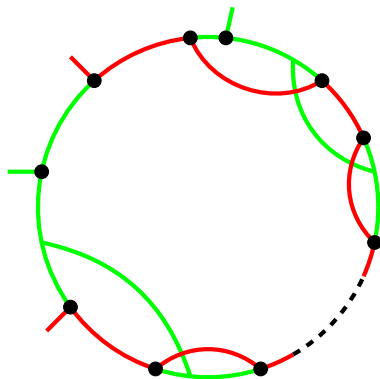
# The Tips: Two Red Edges

- ▶ One path connects to centre.
- ▶ Put it in  $T$  and rest in  $C$ .
- ▶ Problem: “red” chords.
- ▶ Idea: use to shortcut.
- ▶ Must connect new paths.
- ▶ Take maximal sequence of chords such that
  - ▶ paths can be connected,
  - ▶ chords are maximal.



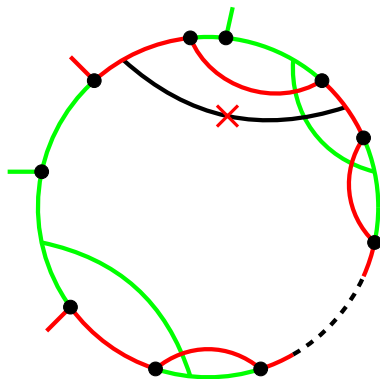
# The Tips: Two Red Edges

- ▶ One path connects to centre.
- ▶ Put it in  $T$  and rest in  $C$ .
- ▶ Problem: “red” chords.
- ▶ Idea: use to shortcut.
- ▶ Must connect new paths.
- ▶ Take maximal sequence of chords such that
  - ▶ paths can be connected,
  - ▶ chords are maximal.
- ▶ Need: No more red chords.



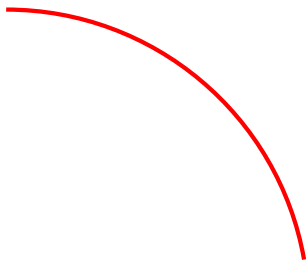
# The Tips: Two Red Edges

- ▶ One path connects to centre.
- ▶ Put it in  $T$  and rest in  $C$ .
- ▶ Problem: “red” chords.
- ▶ Idea: use to shortcut.
- ▶ Must connect new paths.
- ▶ Take maximal sequence of chords such that
  - ▶ paths can be connected,
  - ▶ chords are maximal.
- ▶ Need: No more red chords.
- ▶ By maximality: No chords between different red paths.



# The Tips: Two Red Edges

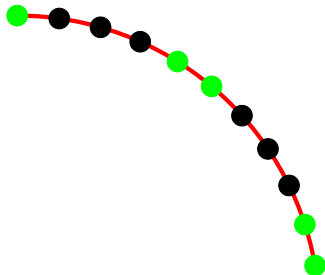
- Call vertices with neighbours in green paths or the centre **good**.





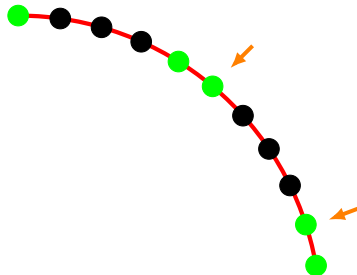
## The Tips: Two Red Edges

- ▶ Call vertices with neighbours in green paths or the centre **good**.
- ▶ Suppose not all vertices in a red path are good.



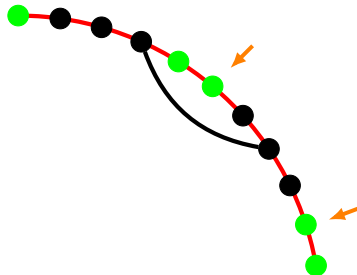
# The Tips: Two Red Edges

- ▶ Call vertices with neighbours in green paths or the centre **good**.
- ▶ Suppose not all vertices in a red path are good.
- ▶ Take two good vertices
  - ▶ with a bad one in between,
  - ▶ of minimal distance.



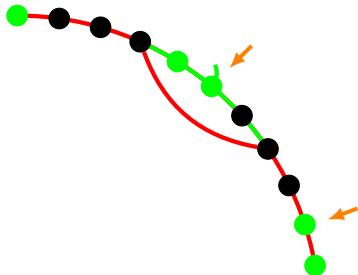
# The Tips: Two Red Edges

- ▶ Call vertices with neighbours in green paths or the centre **good**.
- ▶ Suppose not all vertices in a red path are good.
- ▶ Take two good vertices
  - ▶ with a bad one in between,
  - ▶ of minimal distance.
- ▶ There exists a “crossing” chord.



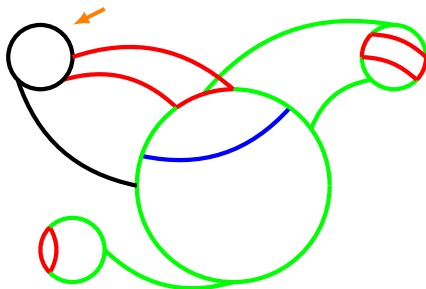
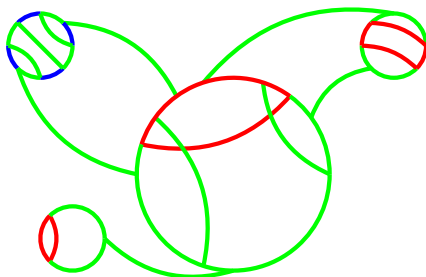
## The Tips: Two Red Edges

- ▶ Call vertices with neighbours in green paths or the centre **good**.
- ▶ Suppose not all vertices in a red path are good.
- ▶ Take two good vertices
  - ▶ with a bad one in between,
  - ▶ of minimal distance.
- ▶ There exists a “crossing” chord.
- ▶ Extends the sequence. ⚡



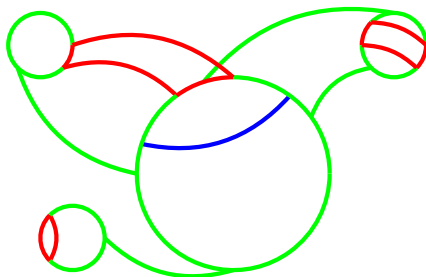
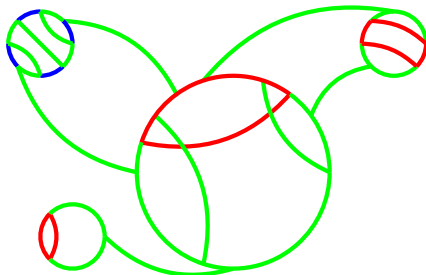
# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
  - ▶ using a part of  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,
  - ▶ 1 green edge,
  - ▶ 2 red edges.



# Decomposing the Centre

- ▶ Take any minimal cycle  $C$ .
- ▶ Check edges to tips.
  - ▶ At most 1 to each tip.
  - ▶ At least 2 to some tip.
- ▶ Use decomposition
  - ▶ given by  $C$ .
  - ▶ using a part of  $C$ .
- ▶ 3 cases for the tips:
  - ▶ only black edges,
  - ▶ 1 green edge,
  - ▶ 2 red edges.



# Summary

- ▶ Regarded a **natural extension** of Hamiltonian graphs.
- ▶ Obtained **reusable decompositions** allowing us to
  - ▶ connect a vertex to the tree,
  - ▶ complete a cycle.
- ▶ Showed why these suffice to decompose **star-like graphs**.

# Summary

- ▶ Regarded a **natural extension** of Hamiltonian graphs.
- ▶ Obtained **reusable decompositions** allowing us to
  - ▶ connect a vertex to the tree,
  - ▶ complete a cycle.
- ▶ Showed why these suffice to decompose **star-like graphs**.



For the sceptics.

Contact: [bachtler@mathematik.uni-kl.de](mailto:bachtler@mathematik.uni-kl.de)



# For Further Reading



S. Akbari, T. R. Jensen, M. Siggers.

Decompositions of graphs into trees, forests, and regular subgraphs.

*Journal of Discrete Mathematics*, vol. 338, no. 8, pp. 1322-1327, 2015.



M. Xie, C. Zhou, S. Zhou.

Decomposition of cubic graphs with a 2-factor consisting of three cycles.

*Journal of Discrete Mathematics*, vol. 343, no. 8, pp. 1118-1139, 2020.



O. Bachtler, S. Krumke.

Towards obtaining a 3-Decomposition from a perfect Matching.

*The Electronic Journal of Combinatorics*, vol. 29, no. 4, 2022.