On 3-Coloring Circle Graphs

GD 2023

Patricia Bachmann, Ignaz Rutter and Peter Stumpf

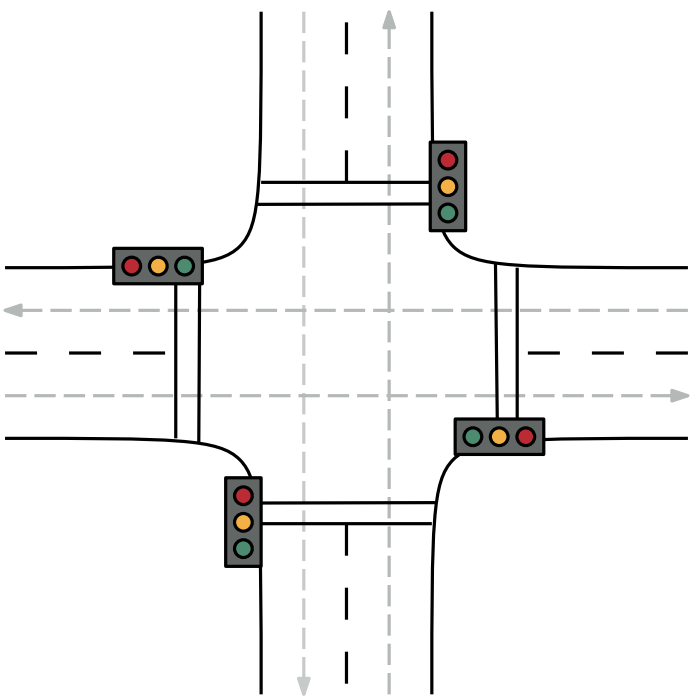
UNIVERSITÄT PASSAU

Fakultät für Informatik und Mathematik

Introduction

k-Page Book Embedding

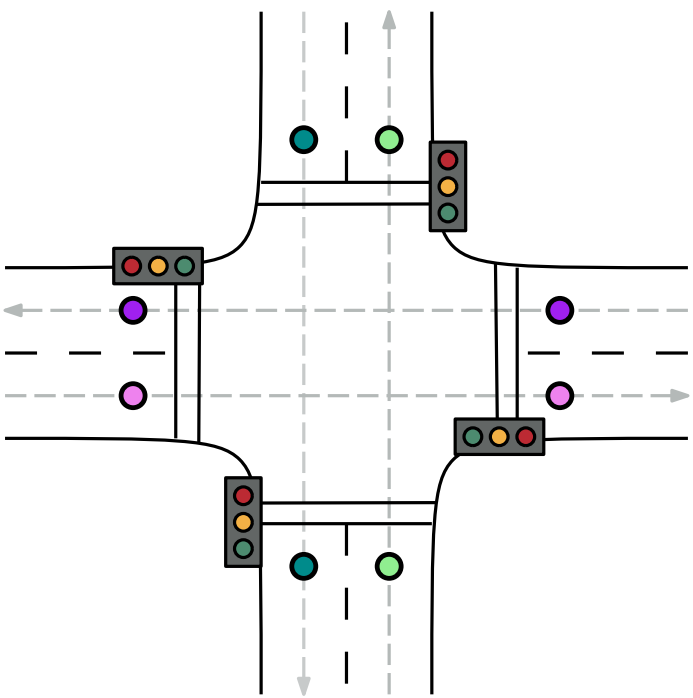
- partition of edges into k pages
- no page contains crossing edges



Introduction

k-Page Book Embedding

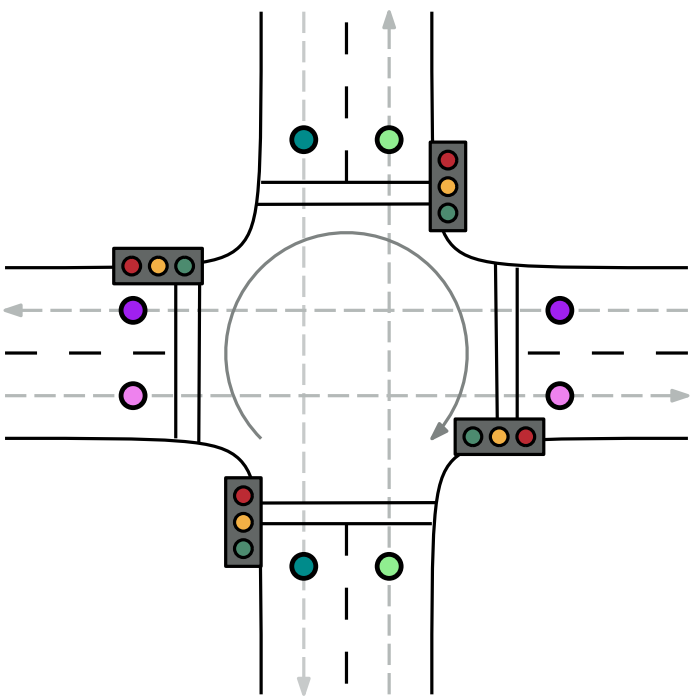
- partition of edges into k pages
- no page contains crossing edges



Introduction

k-Page Book Embedding

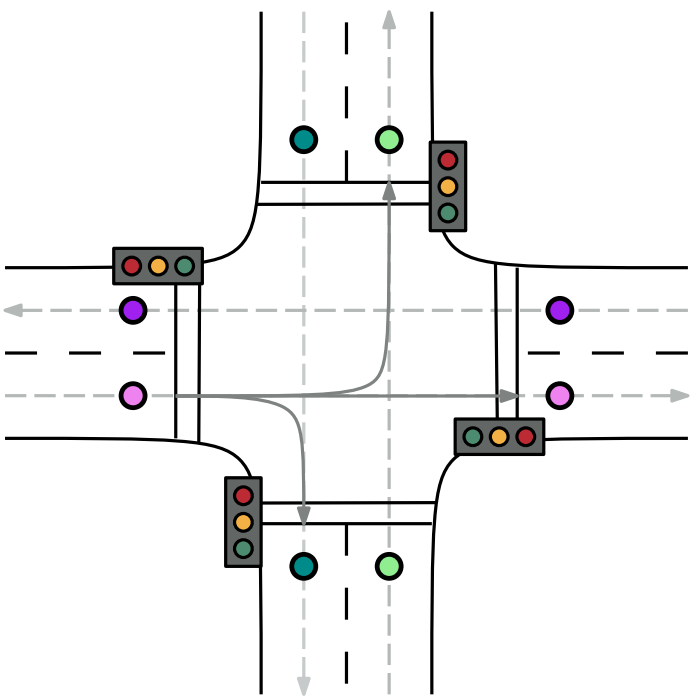
- partition of edges into k pages
- no page contains crossing edges

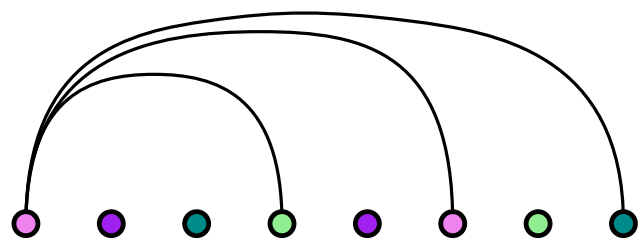


Introduction

k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges

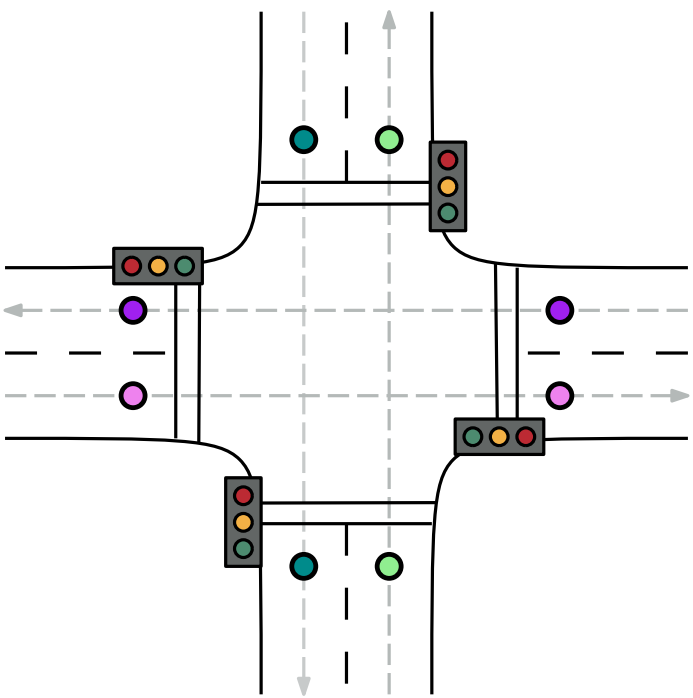


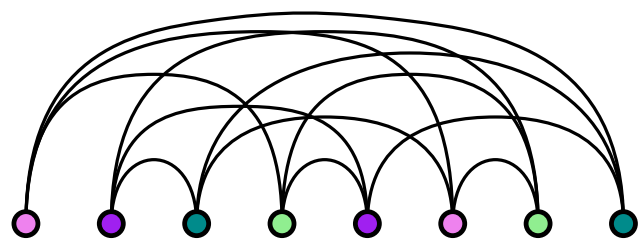


Introduction

k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges

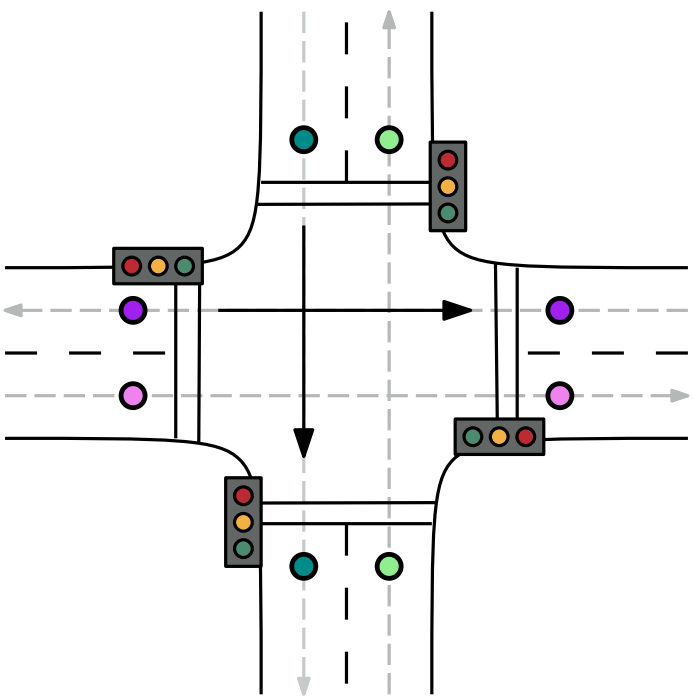


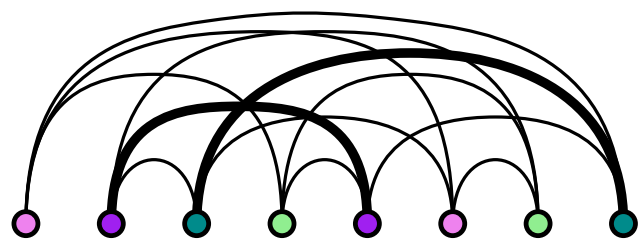


Introduction

k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges

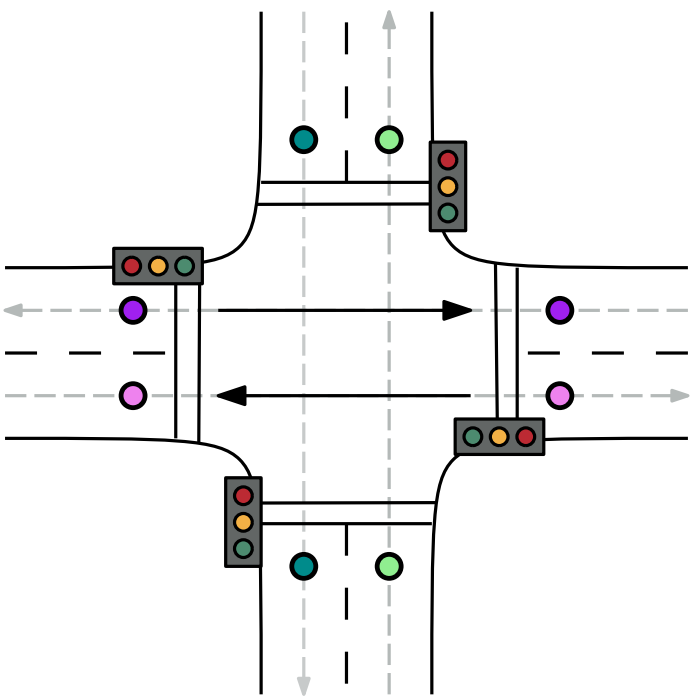


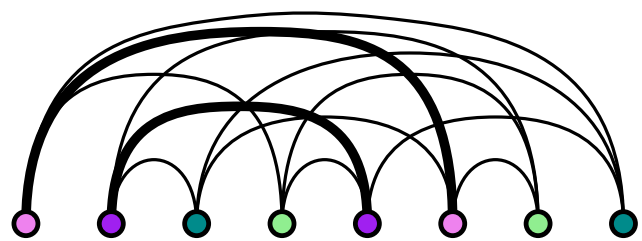


Introduction

k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges

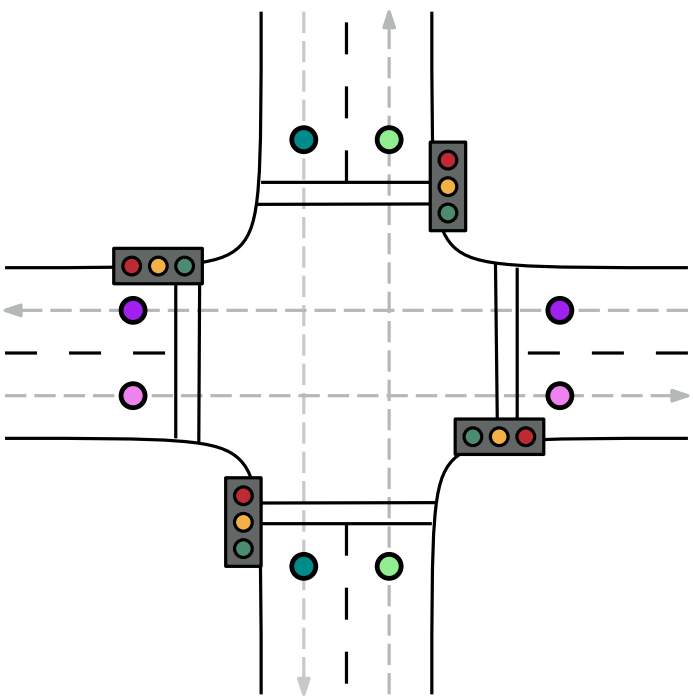


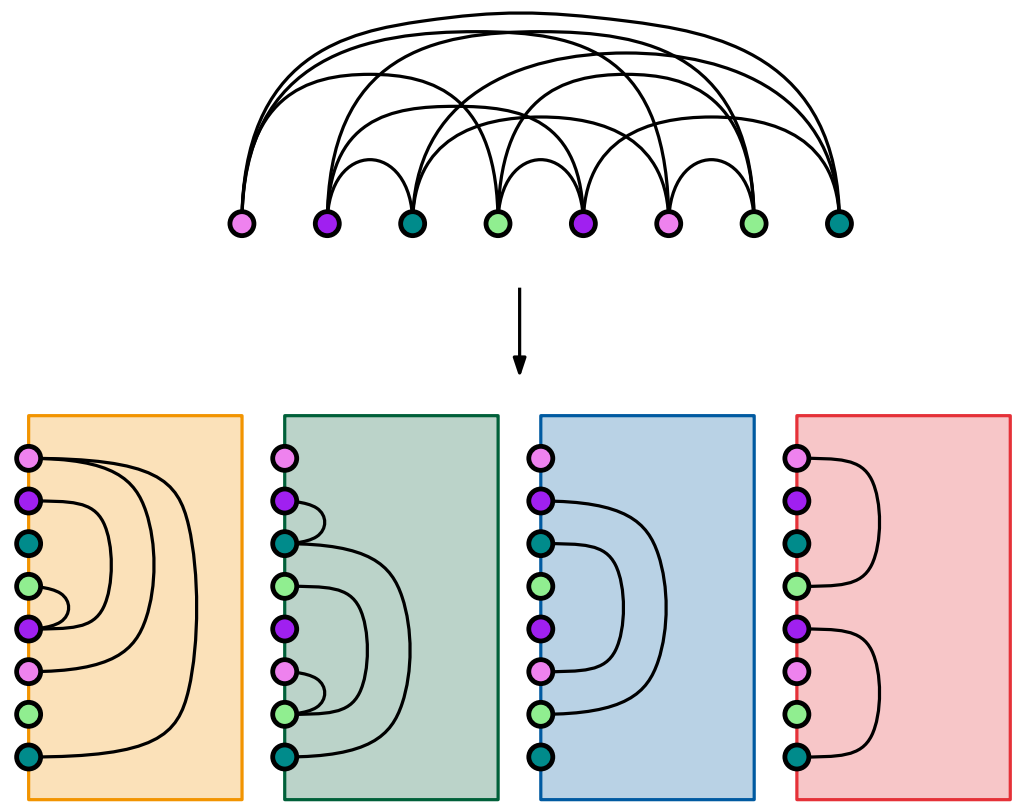


Introduction

k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges



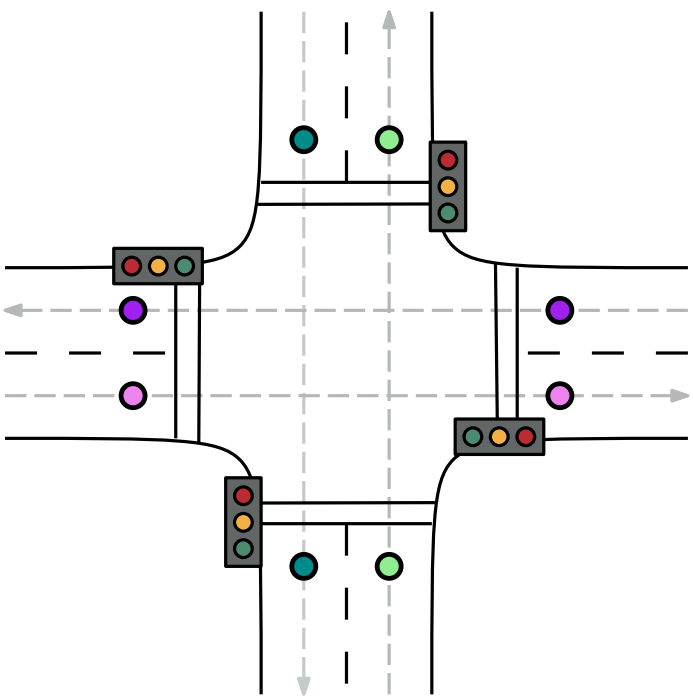


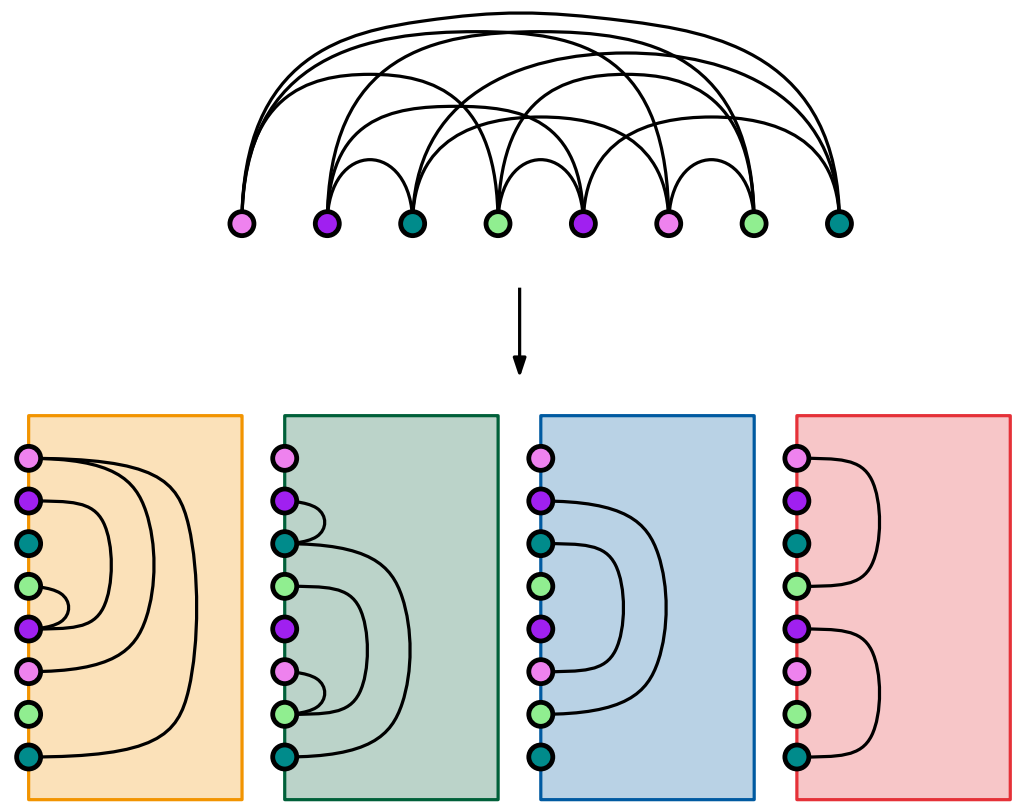
Introduction

k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges

- VLSI design [Chung, Leighton, Rosenberg ’87]
- arc diagrams [Wattenberg ’87]
- circular layouts [Baur, Brandes ’04]
- clustered planarity [Hong, Nagamochi ’18]
- simultaneous embedding
  [Angelini, de Battista, Frati, Patrignani, Rutter ’12]

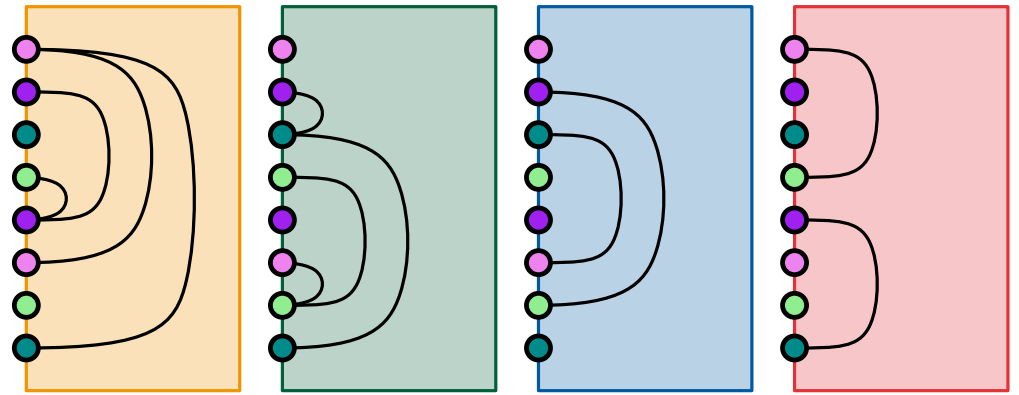


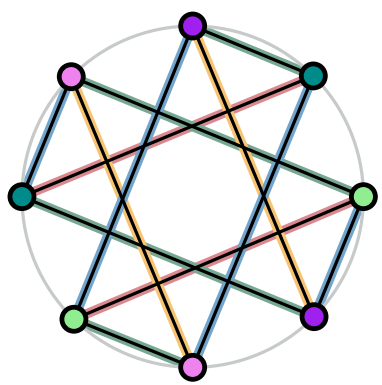


Introduction

k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges

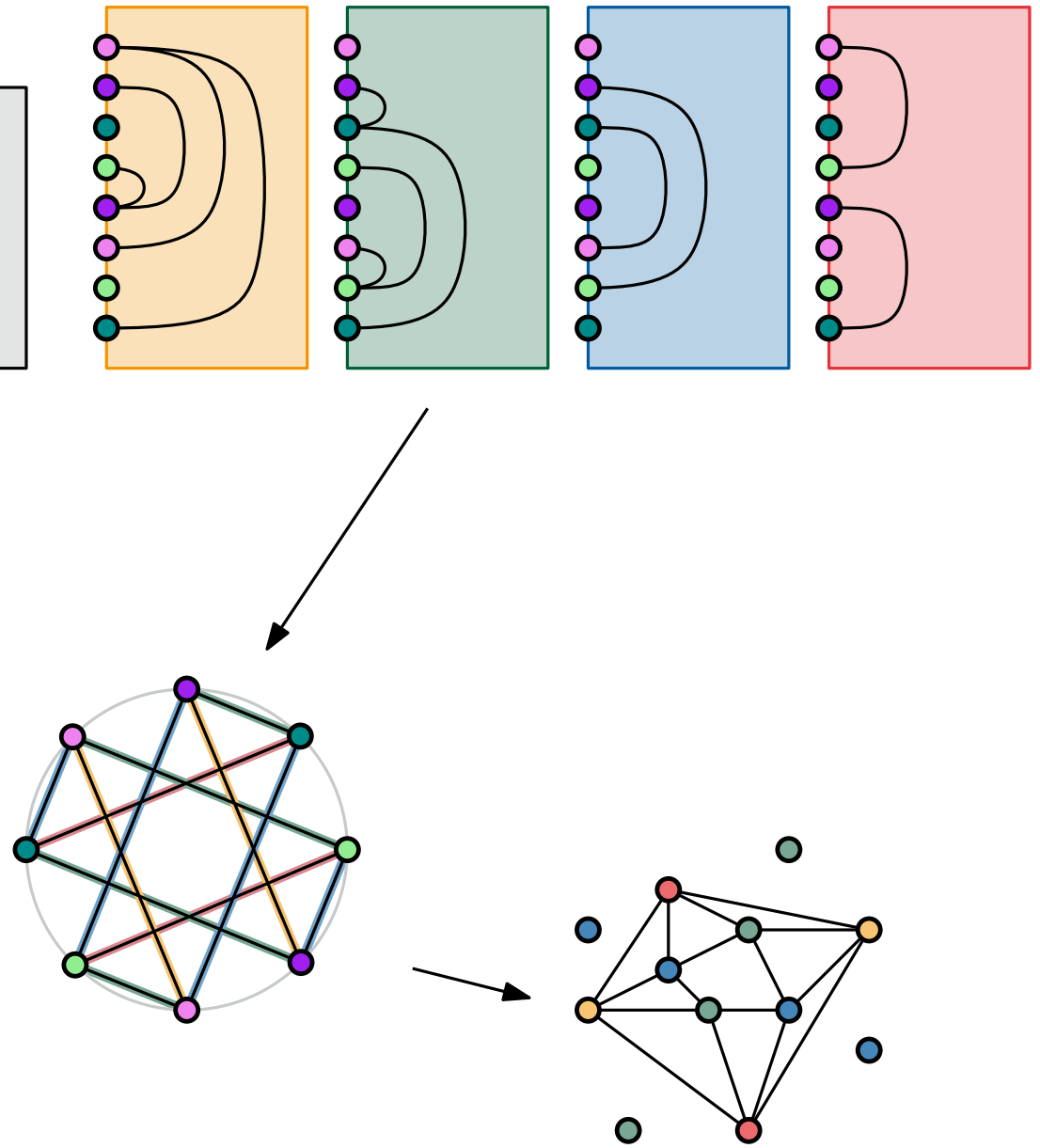




Introduction

k-Page Book Embedding

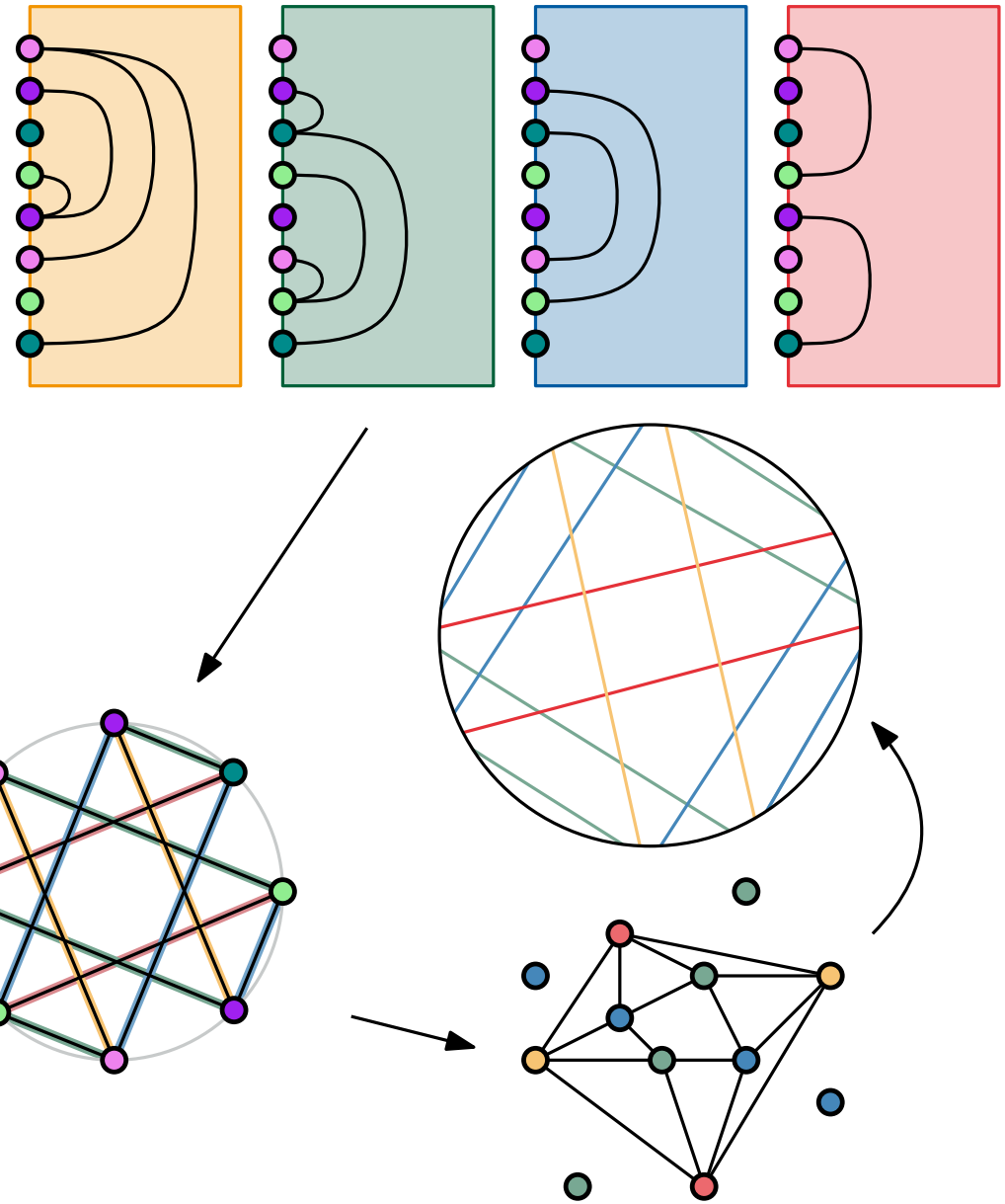
- partition of edges into k pages
- no page contains crossing edges



Introduction

k-Page Book Embedding

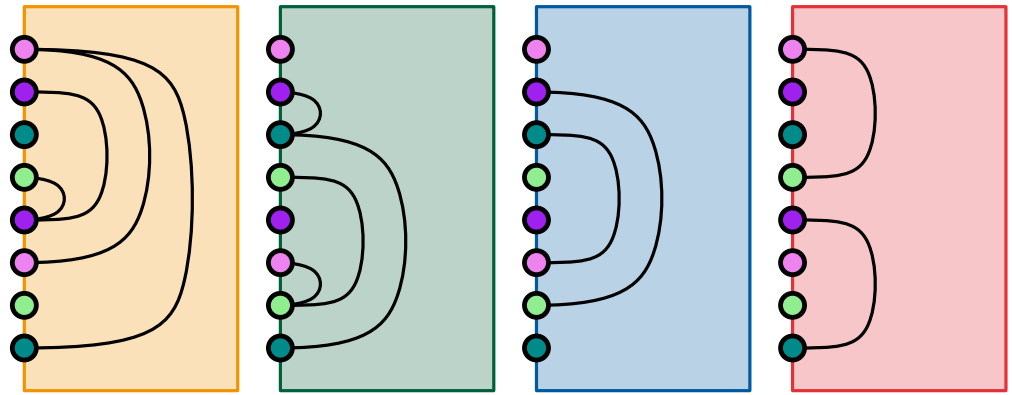
- partition of edges into k pages
- no page contains crossing edges



Introduction

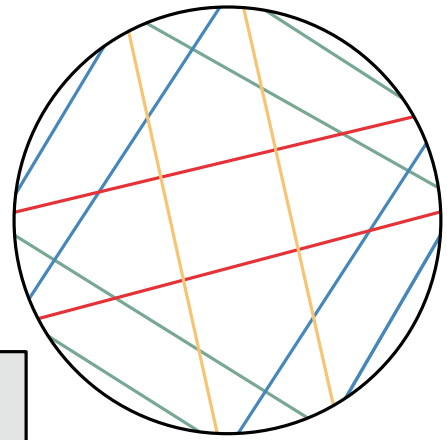
k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges



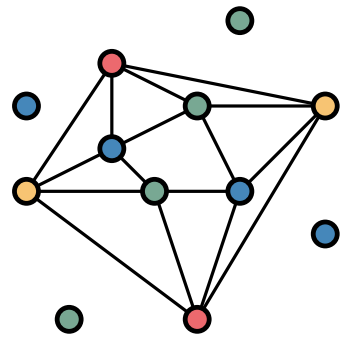
Circle Graph

- vertices of G represented as chords on a circle
- two chords cross $\Leftrightarrow$ vertices are adjacent



Coloring

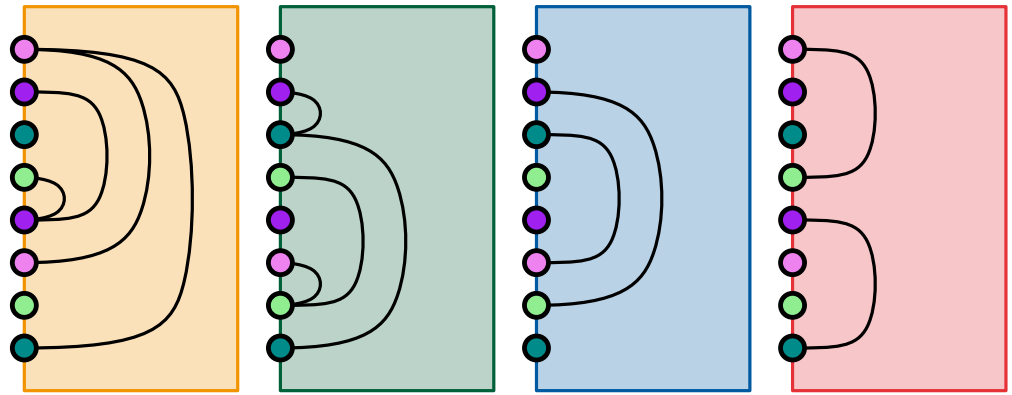
- color a graph G: adjacent vertices have a different color
- find smallest number of colors s.t. G can be colored



Introduction

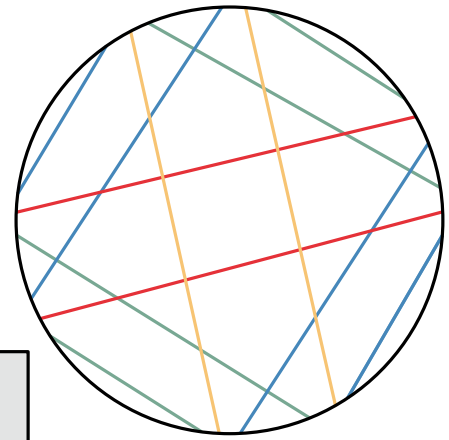
k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges



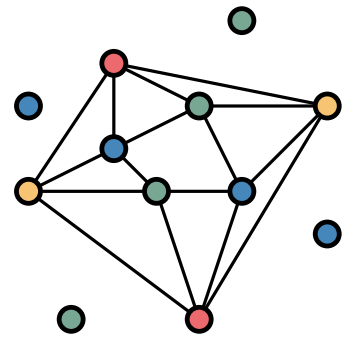
Circle Graph

- vertices of G represented as chords on a circle
- two chords cross $\Leftrightarrow$ vertices are adjacent



k-Coloring

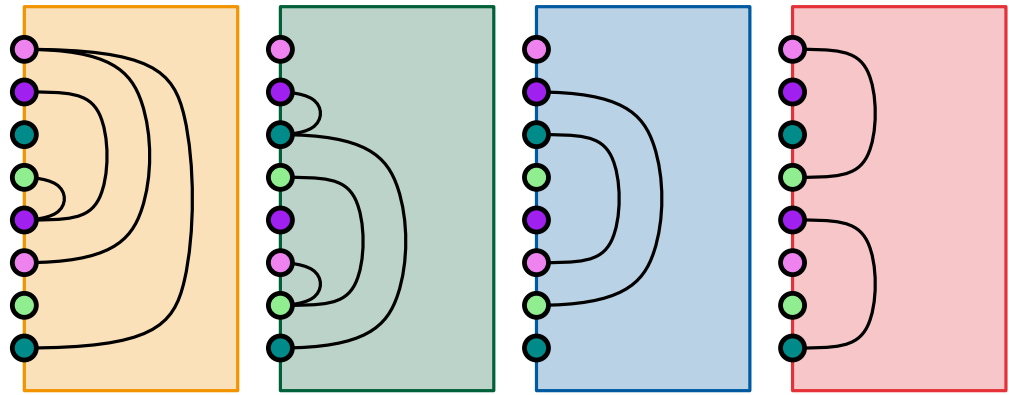
- color a graph G: adjacent vertices have a different color
- can G be colored using at most k colors?



Introduction

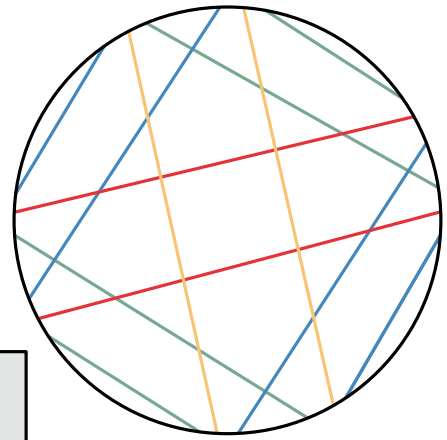
k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges



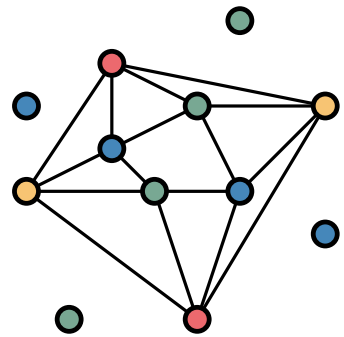
Circle Graph

- vertices of G represented as chords on a circle
- two chords cross $\Leftrightarrow$ vertices are adjacent



k-Coloring $\rightarrow$ NP-complete (for $k \geq 3$)

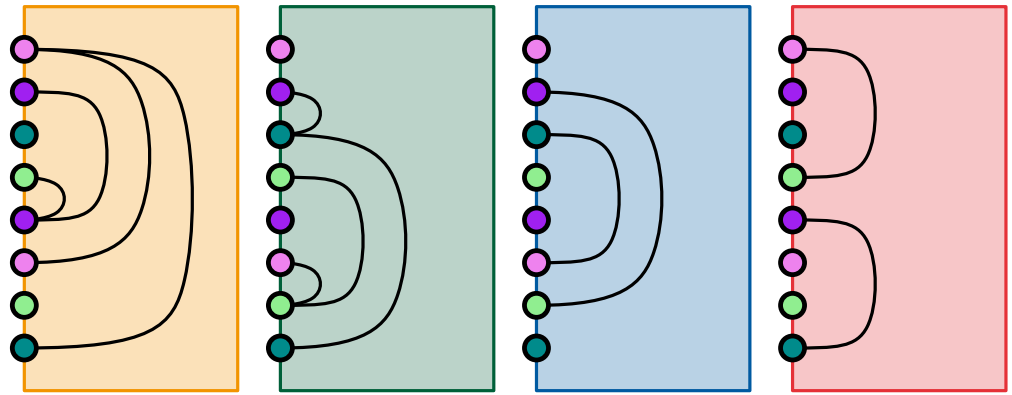
- color a graph G: adjacent vertices have a different color
- can G be colored using at most k colors?



Introduction

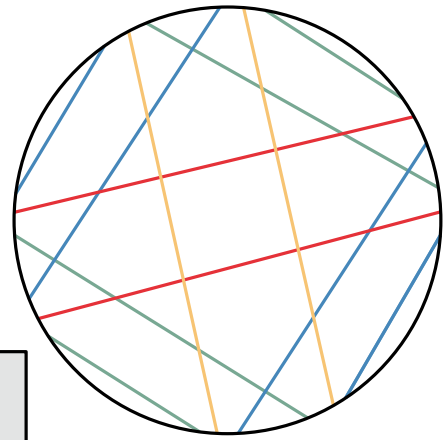
k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges



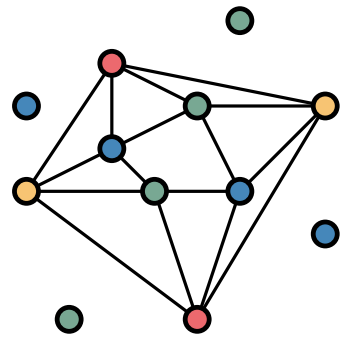
Circle Graph

- vertices of G represented as chords on a circle
- two chords cross $\Leftrightarrow$ vertices are adjacent



k-Coloring $\rightarrow$ NP-complete (for $k \geq 3$)

- color a graph G: adjacent vertices have a different color
- can G be colored using at most k colors?



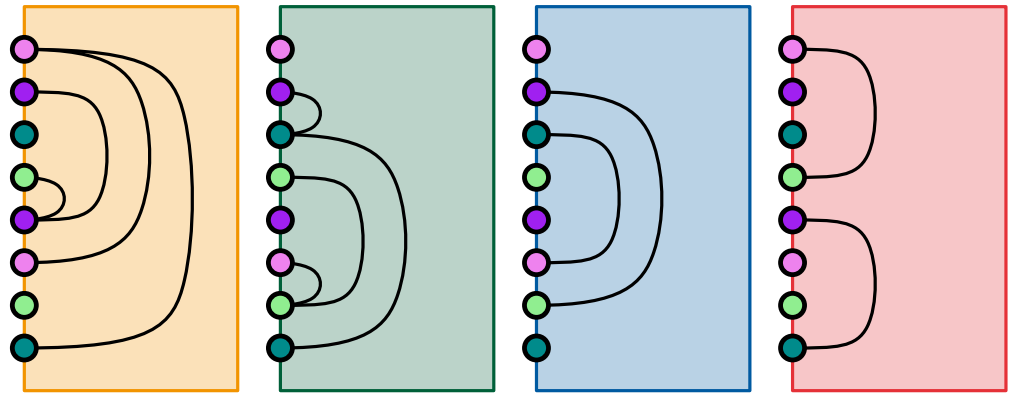
Unger (1990):

- 4-coloring problem NP-complete for circle graphs

Introduction

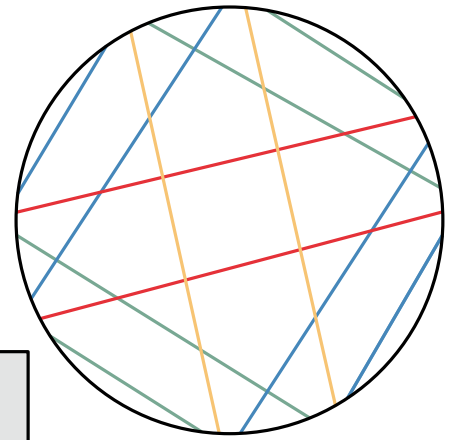
k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges



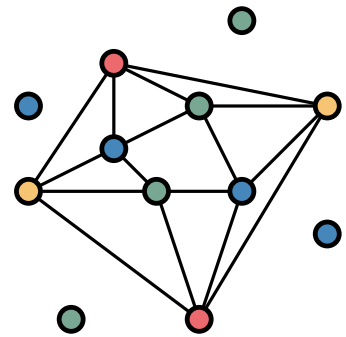
Circle Graph

- vertices of G represented as chords on a circle
- two chords cross $\Leftrightarrow$ vertices are adjacent



k-Coloring $\rightarrow$ NP-complete (for $k \geq 3$)

- color a graph G: adjacent vertices have a different color
- can G be colored using at most k colors?



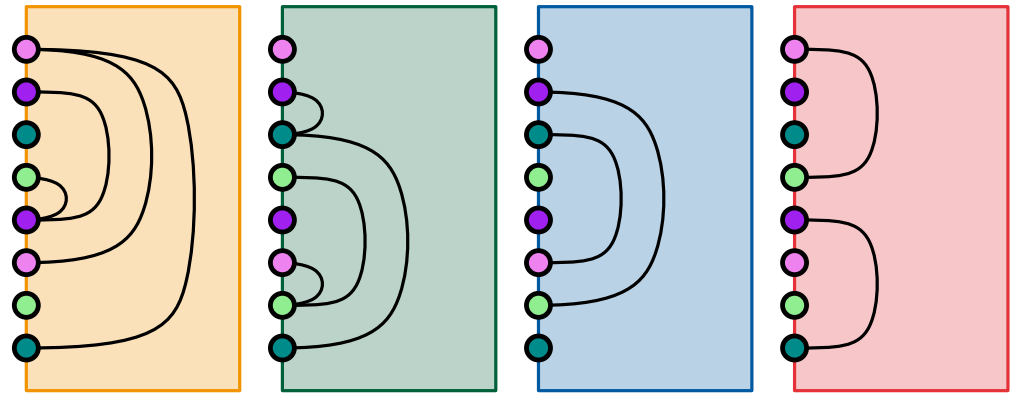
Unger (1990):

- 4-coloring problem NP-complete for circle graphs
- claimed 3-coloring solvable in $O(n \log n)$ time for circle graphs

Introduction

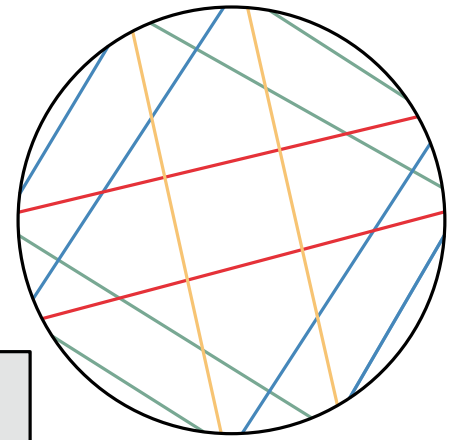
k-Page Book Embedding

- partition of edges into k pages
- no page contains crossing edges



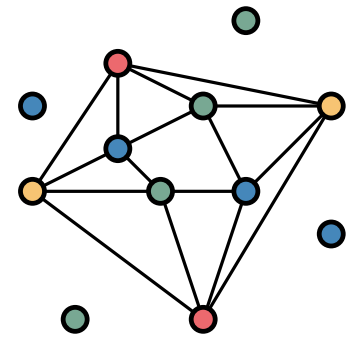
Circle Graph

- vertices of G represented as chords on a circle
- two chords cross $\Leftrightarrow$ vertices are adjacent



k-Coloring $\rightarrow$ NP-complete (for $k \geq 3$)

- color a graph G: adjacent vertices have a different color
- can G be colored using at most k colors?



Unger (1990):

- 4-coloring problem NP-complete for circle graphs
- claimed 3-coloring solvable in $O(n \log n)$ time for circle graphs

  $\rightarrow$ proofs for 3-coloring were never published in an archival paper, only in his PhD Thesis

  $\rightarrow$ unfortunate state of affairs pointed out by David Eppstein and Dujmović, Pór, Wood

Outline

3-coloring Circle Graphs

Outline

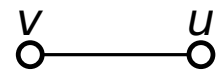
3-coloring Circle Graphs

2-SAT

Outline

3-coloring Circle Graphs

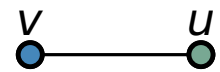
2-SAT



Outline

3-coloring Circle Graphs

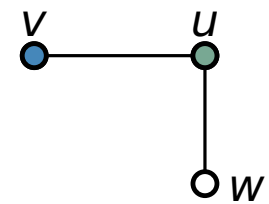
2-SAT



Outline

3-coloring Circle Graphs

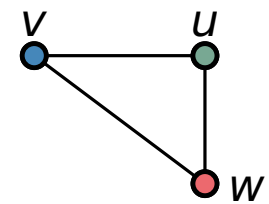
2-SAT



Outline

3-coloring Circle Graphs

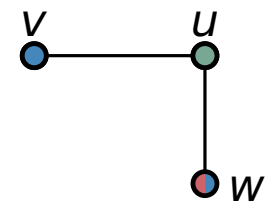
2-SAT



Outline

3-coloring Circle Graphs

2-SAT

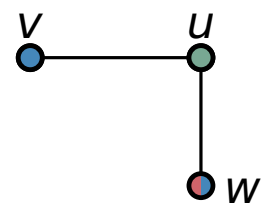


Outline

3-coloring Circle Graphs

2-SAT

- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$

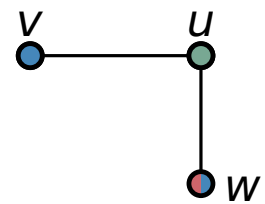


Outline

3-coloring Circle Graphs

2-SAT

- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$ $\subseteq \binom{V(G)}{2}$, common neighbor but not adjacent

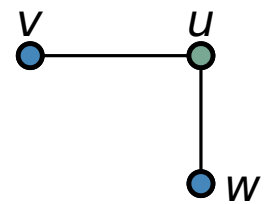


Outline

3-coloring Circle Graphs

2-SAT

- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$ $\subseteq \binom{V(G)}{2}$, common neighbor but not adjacent



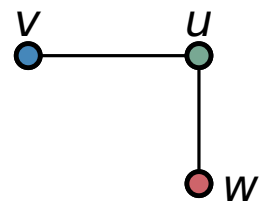
$c_{\text{aux}}(v, w) = \text{true}$

Outline

3-coloring Circle Graphs

2-SAT

- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$ $\subseteq \binom{V(G)}{2}$, common neighbor but not adjacent



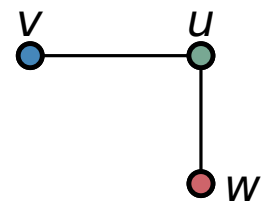
$c_{\text{aux}}(v, w) = \text{false}$

Outline

3-coloring Circle Graphs

2-SAT

- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$ $\subseteq \binom{V(G)}{2}$, common neighbor but not adjacent
- assuming G connected and c_{aux} "correct" $\Rightarrow$ unique 3-coloring



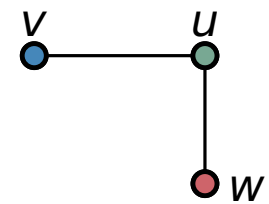
$c_{\text{aux}}(v, w) = \text{false}$

Outline

3-coloring Circle Graphs

2-SAT

- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$ ($\mathcal{D} \subseteq \binom{V(G)}{2}$, common neighbor but not adjacent)
- assuming G connected and c_{aux} "correct"? $\Rightarrow$ unique 3-coloring



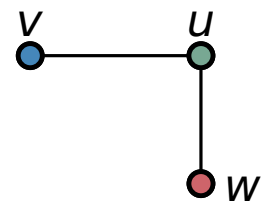
$c_{\text{aux}}(v, w) = \text{false}$

Outline

3-coloring Circle Graphs

2-SAT

- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$ $\mathcal{D} \subseteq \binom{V(G)}{2}$, common neighbor but not adjacent
- assuming G connected and c_{aux} "correct" $\Rightarrow$ unique 3-coloring
- constraints for c_{aux} ensure 3-coloring for subgraphs



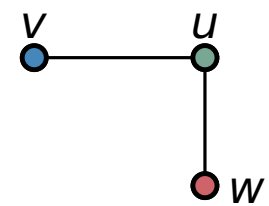
$c_{\text{aux}}(v, w) = \text{false}$

Outline

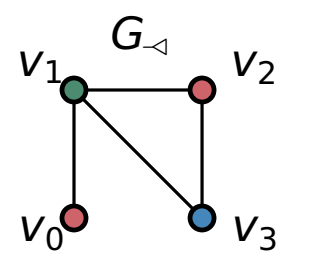
3-coloring Circle Graphs

2-SAT

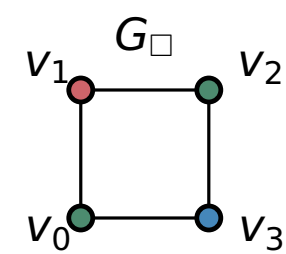
- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$ ($\mathcal{D} \subseteq \binom{V(G)}{2}$, common neighbor but not adjacent)
- assuming G connected and c_{aux} "correct" $\Rightarrow$ unique 3-coloring
- constraints for c_{aux} ensure 3-coloring for subgraphs



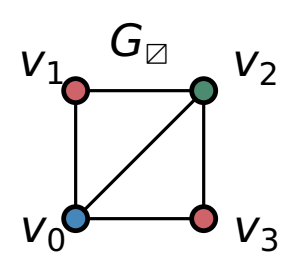
$c_{\text{aux}}(v, w) = \text{false}$



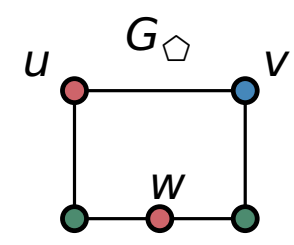
$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



$v_0v_2 \vee v_1v_3$



v_1v_3



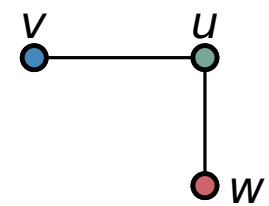
$\neg uw \vee \neg vw$

Outline

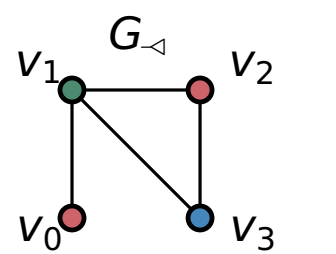
3-coloring Circle Graphs

2-SAT

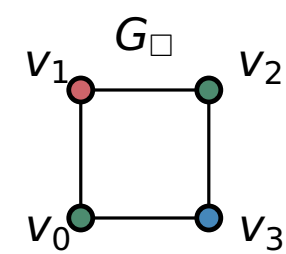
- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$ $\subseteq \binom{V(G)}{2}$, common neighbor but not adjacent
- assuming G connected and c_{aux} "correct" $\Rightarrow$ unique 3-coloring
- constraints for c_{aux} ensure 3-coloring for subgraphs
- solve 2-SAT instance to get values for c^*



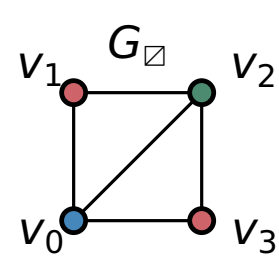
$c_{\text{aux}}(v, w) = \text{false}$



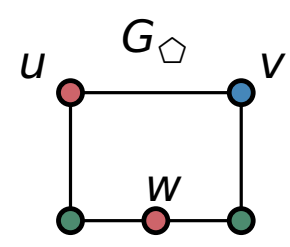
$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



$v_0v_2 \vee v_1v_3$



v_1v_3



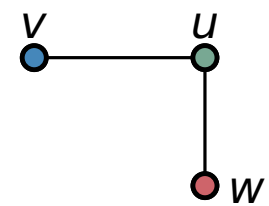
$\neg uw \vee \neg vw$

Outline

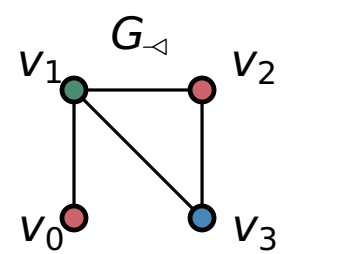
3-coloring Circle Graphs

2-SAT

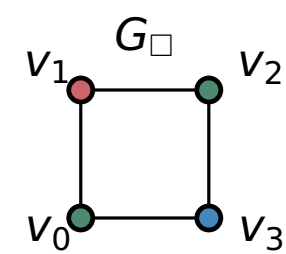
- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$ ($\mathcal{D} \subseteq \binom{V(G)}{2}$, common neighbor but not adjacent)
- assuming G connected and c_{aux} "correct" $\Rightarrow$ unique 3-coloring
- constraints for c_{aux} ensure 3-coloring for subgraphs
- solve 2-SAT instance to get values for c^*
- use backtracking to compute c_{aux} for cycles of length ≥ 5



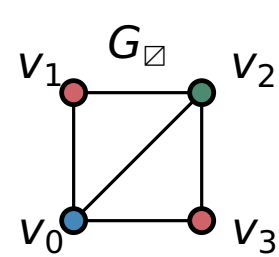
$c_{\text{aux}}(v, w) = \text{false}$



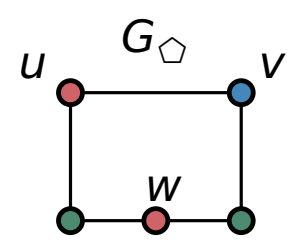
$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



$v_0v_2 \vee v_1v_3$



v_1v_3



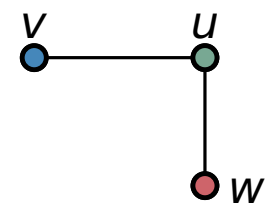
$\neg uw \vee \neg vw$

Outline

3-coloring Circle Graphs

2-SAT

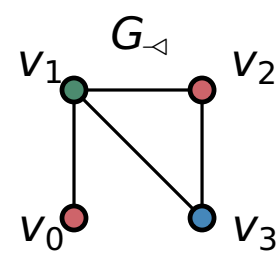
- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$ ($\mathcal{D} \subseteq \binom{V(G)}{2}$, common neighbor but not adjacent)
- assuming G connected and c_{aux} "correct" $\Rightarrow$ unique 3-coloring
- constraints for c_{aux} ensure 3-coloring for subgraphs
- solve 2-SAT instance to get values for c^*
- use backtracking to compute c_{aux} for cycles of length ≥ 5



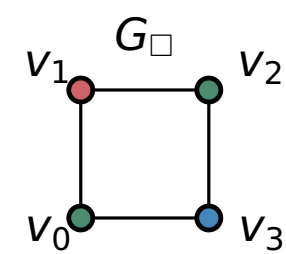
$c_{\text{aux}}(v, w) = \text{false}$

Unger claims:

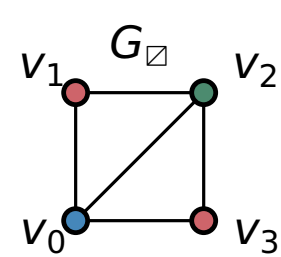
- SAT formula is satisfiable $\Leftrightarrow$ graph is 3-colorable



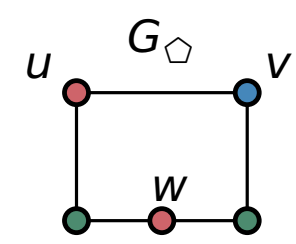
$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



$v_0v_2 \vee v_1v_3$



v_1v_3



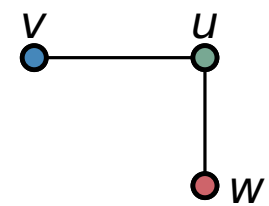
$\neg uw \vee \neg vw$

Outline

3-coloring Circle Graphs

2-SAT

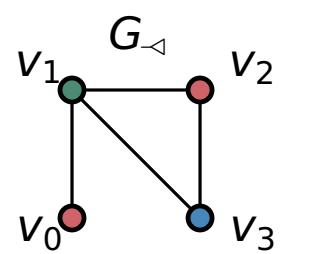
- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$ ($\mathcal{D} \subseteq \binom{V(G)}{2}$, common neighbor but not adjacent)
- assuming G connected and c_{aux} "correct" $\Rightarrow$ unique 3-coloring
- constraints for c_{aux} ensure 3-coloring for subgraphs
- solve 2-SAT instance to get values for c^*
- use backtracking to compute c_{aux} for cycles of length ≥ 5



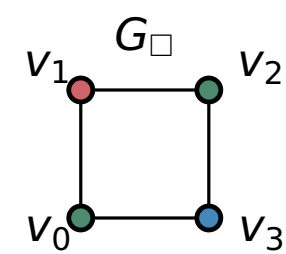
$c_{\text{aux}}(v, w) = \text{false}$

Unger claims:

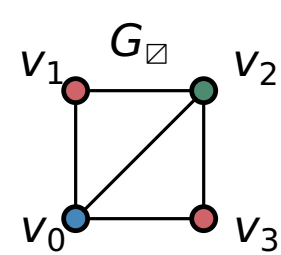
- SAT formula is satisfiable $\Leftrightarrow$ graph is 3-colorable
- backtracking tree has $O(\log n)$ leaves



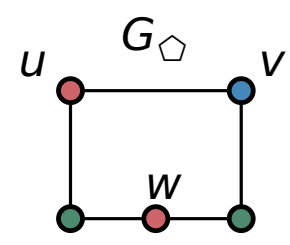
$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



$v_0v_2 \vee v_1v_3$



v_1v_3



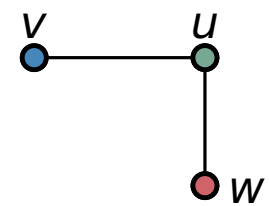
$\neg uw \vee \neg vw$

Outline

3-coloring Circle Graphs

2-SAT

- define function $c_{\text{aux}} : \mathcal{D} \to \{\text{true}, \text{false}\}$ ($\mathcal{D} \subseteq \binom{V(G)}{2}$, common neighbor but not adjacent)
- assuming G connected and c_{aux} "correct" $\Rightarrow$ unique 3-coloring
- constraints for c_{aux} ensure 3-coloring for subgraphs
- solve 2-SAT instance to get values for c^*
- use backtracking to compute c_{aux} for cycles of length ≥ 5

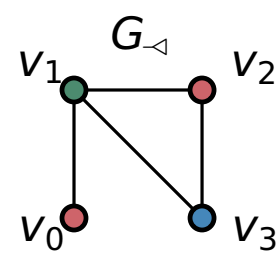


$c_{\text{aux}}(v, w) = \text{false}$

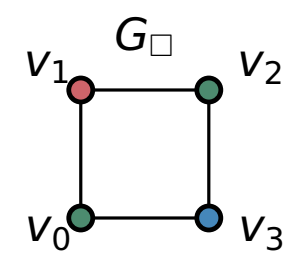
Unger claims:

- SAT formula is satisfiable $\Leftrightarrow$ graph is 3-colorable
- backtracking tree has $O(\log n)$ leaves

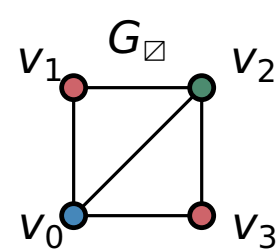
Our contribution: counterexamples to both claims



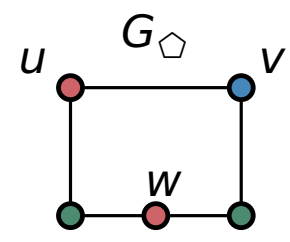
$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



$v_0v_2 \vee v_1v_3$



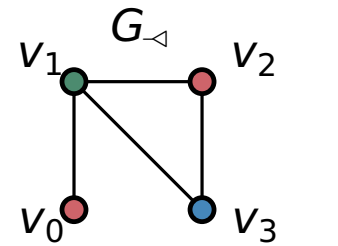
v_1v_3



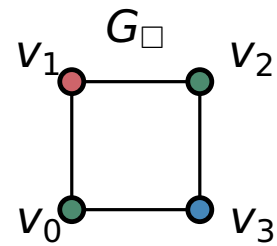
$\neg uw \vee \neg vw$

First Counterexample

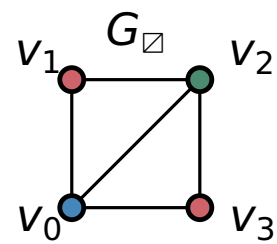
Claim 1. SAT formula is satisfiable ⇔ graph is 3-colorable



$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



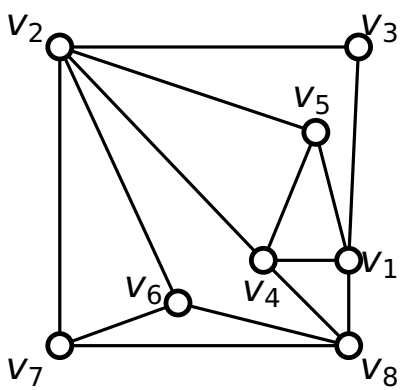
$v_0v_2 \vee v_1v_3$

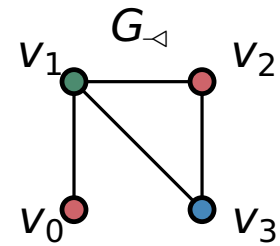


v_1v_3

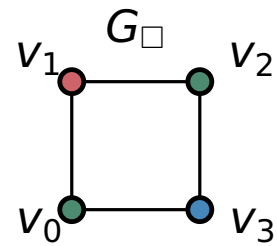
First Counterexample

Claim 1. SAT formula is satisfiable ⇔ graph is 3-colorable

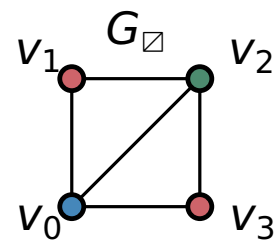




$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



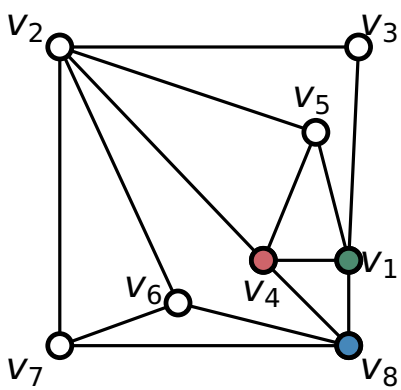
$v_0v_2 \vee v_1v_3$

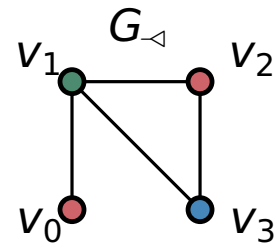


v_1v_3

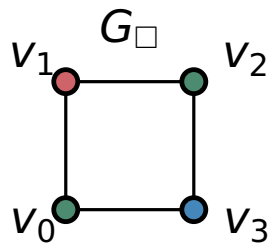
First Counterexample

Claim 1. SAT formula is satisfiable ⇔ graph is 3-colorable

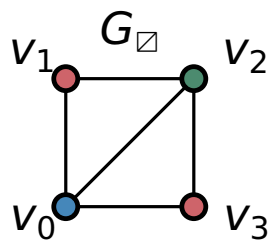




$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



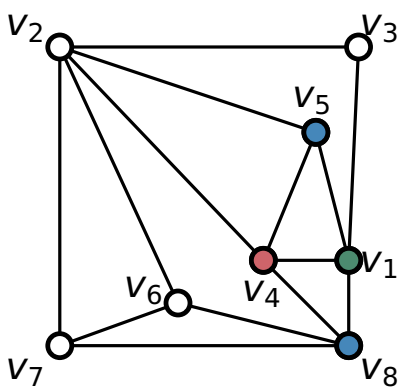
$v_0v_2 \vee v_1v_3$

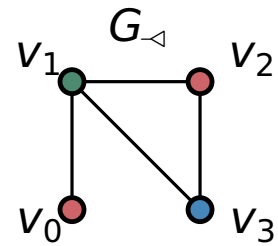


v_1v_3

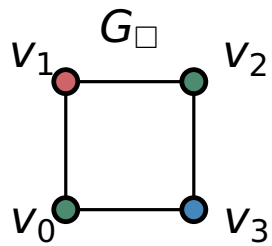
First Counterexample

Claim 1. SAT formula is satisfiable ⇔ graph is 3-colorable

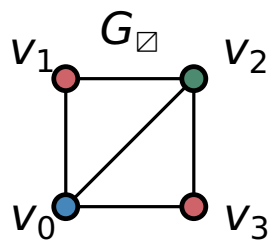




$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



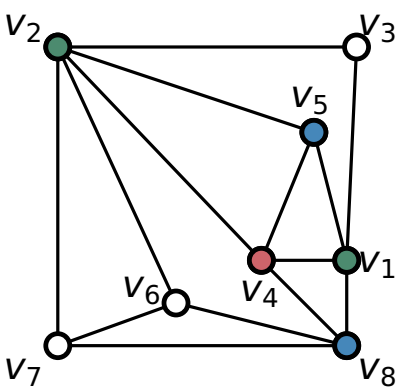
$v_0v_2 \vee v_1v_3$

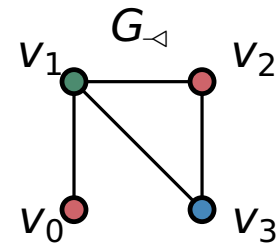


v_1v_3

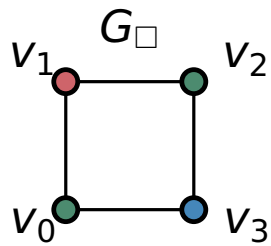
First Counterexample

Claim 1. SAT formula is satisfiable ⇔ graph is 3-colorable

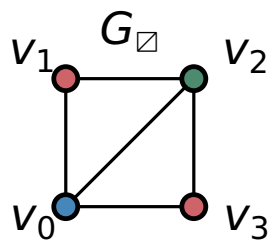




$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



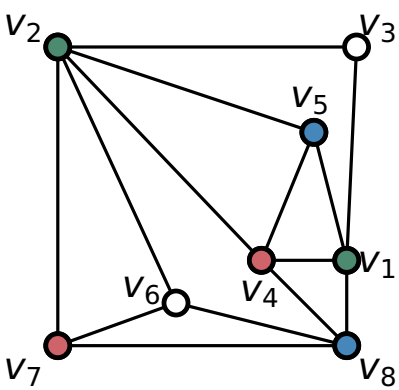
$v_0v_2 \vee v_1v_3$

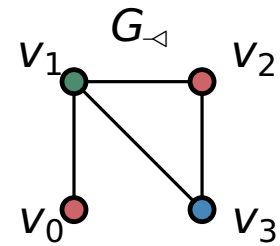


v_1v_3

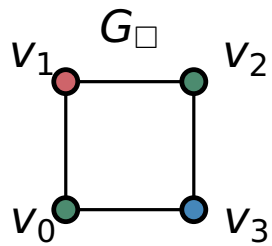
First Counterexample

Claim 1. SAT formula is satisfiable ⇔ graph is 3-colorable

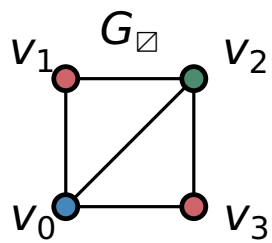




$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



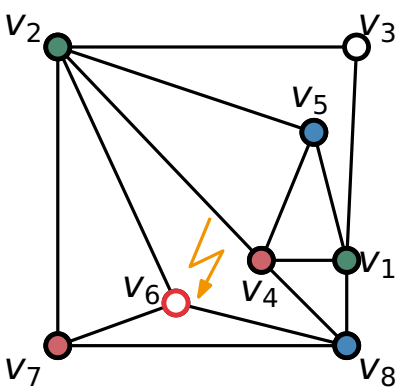
$v_0v_2 \vee v_1v_3$

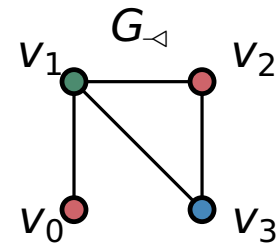


v_1v_3

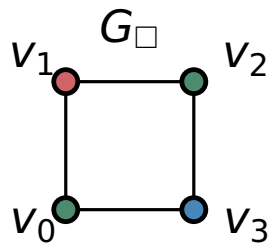
First Counterexample

Claim 1. SAT formula is satisfiable ⇔ graph is 3-colorable

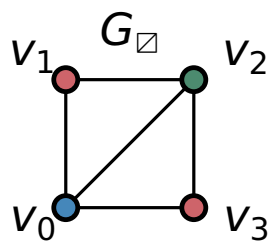




$(v_0v_2 \vee v_0v_3) \wedge (\neg v_0v_2 \vee \neg v_0v_3)$



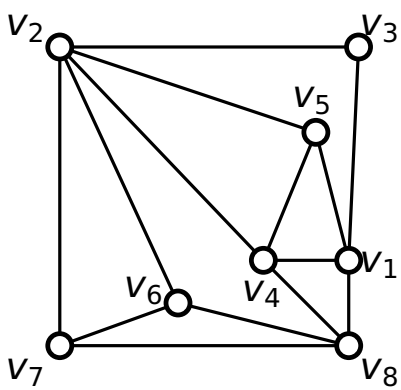
$v_0v_2 \vee v_1v_3$



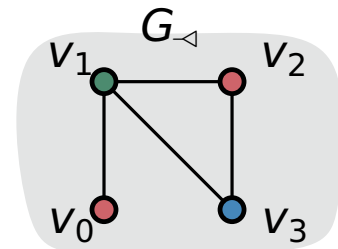
v_1v_3

First Counterexample

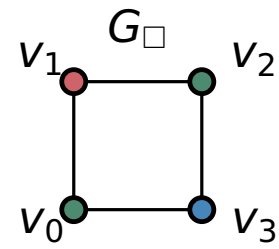
Claim 1. SAT formula is satisfiable ⇔ graph is 3-colorable



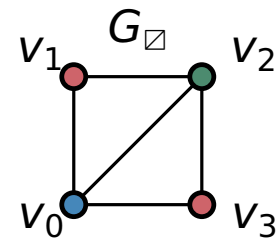
$$(v_3v_4 \vee v_3v_5) \wedge (\neg v_3v_4 \vee \neg v_3v_5) \wedge$$
$$(v_1v_6 \vee v_1v_7) \wedge (\neg v_1v_6 \vee \neg v_1v_7) \wedge$$
$$(v_4v_6 \vee v_4v_7) \wedge (\neg v_4v_6 \vee \neg v_4v_7) \wedge$$
$$(v_5v_6 \vee v_5v_7) \wedge (\neg v_5v_6 \vee \neg v_5v_7) \wedge$$
$$(v_5v_6 \vee v_5v_7) \wedge (\neg v_5v_6 \vee \neg v_5v_7) \wedge$$
$$(v_4v_7 \vee v_5v_7) \wedge (\neg v_4v_7 \vee \neg v_5v_7) \wedge$$



$(v_0v_2 \vee v_0v_3) \wedge$
$(\neg v_0v_2 \vee \neg v_0v_3)$



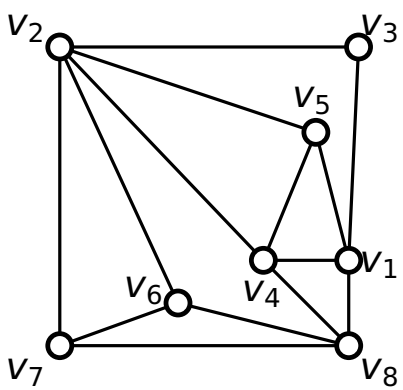
$v_0v_2 \vee v_1v_3$



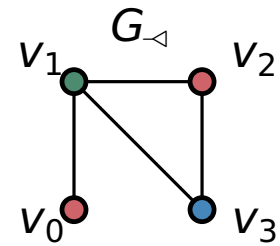
v_1v_3

First Counterexample

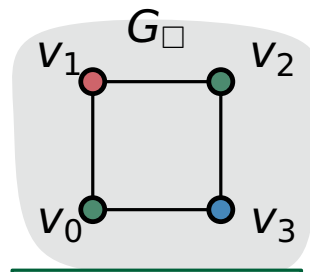
Claim 1. SAT formula is satisfiable ⇔ graph is 3-colorable



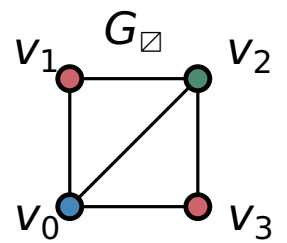
$$(v_3v_4 \vee v_3v_5) \wedge (\neg v_3v_4 \vee \neg v_3v_5) \wedge$$
$$(v_1v_6 \vee v_1v_7) \wedge (\neg v_1v_6 \vee \neg v_1v_7) \wedge$$
$$(v_4v_6 \vee v_4v_7) \wedge (\neg v_4v_6 \vee \neg v_4v_7) \wedge$$
$$(v_5v_6 \vee v_5v_7) \wedge (\neg v_5v_6 \vee \neg v_5v_7) \wedge$$
$$(v_5v_6 \vee v_5v_7) \wedge (\neg v_5v_6 \vee \neg v_5v_7) \wedge$$
$$(v_4v_7 \vee v_5v_7) \wedge (\neg v_4v_7 \vee \neg v_5v_7) \wedge$$
$$(v_1v_2 \vee v_3v_4) \wedge$$
$$(v_1v_2 \vee v_3v_5) \wedge$$



$$(v_0v_2 \vee v_0v_3) \wedge$$
$$(\neg v_0v_2 \vee \neg v_0v_3)$$



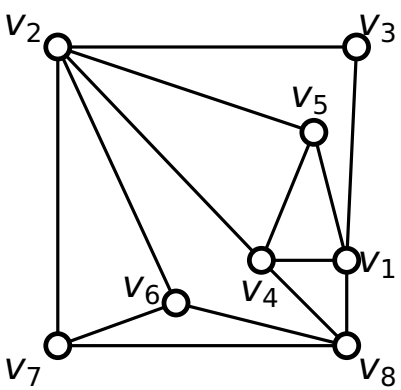
$$v_0v_2 \vee v_1v_3$$



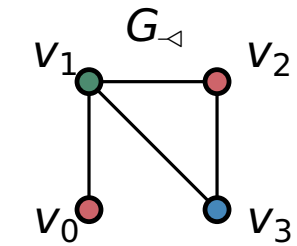
$$v_1v_3$$

First Counterexample

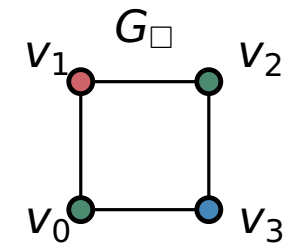
Claim 1. SAT formula is satisfiable ⇔ graph is 3-colorable



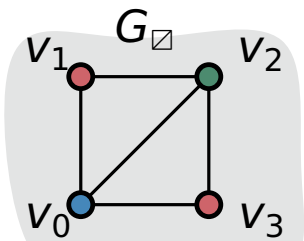
$$(v_3v_4 \vee v_3v_5) \wedge (\neg v_3v_4 \vee \neg v_3v_5) \wedge$$
$$(v_1v_6 \vee v_1v_7) \wedge (\neg v_1v_6 \vee \neg v_1v_7) \wedge$$
$$(v_4v_6 \vee v_4v_7) \wedge (\neg v_4v_6 \vee \neg v_4v_7) \wedge$$
$$(v_5v_6 \vee v_5v_7) \wedge (\neg v_5v_6 \vee \neg v_5v_7) \wedge$$
$$(v_5v_6 \vee v_5v_7) \wedge (\neg v_5v_6 \vee \neg v_5v_7) \wedge$$
$$(v_4v_7 \vee v_5v_7) \wedge (\neg v_4v_7 \vee \neg v_5v_7) \wedge$$
$$(v_1v_2 \vee v_3v_4) \wedge$$
$$(v_1v_2 \vee v_3v_5) \wedge$$
$$(v_1v_2) \wedge$$
$$(v_5v_8)$$



$(v_0v_2 \vee v_0v_3) \wedge$
$(\neg v_0v_2 \vee \neg v_0v_3)$



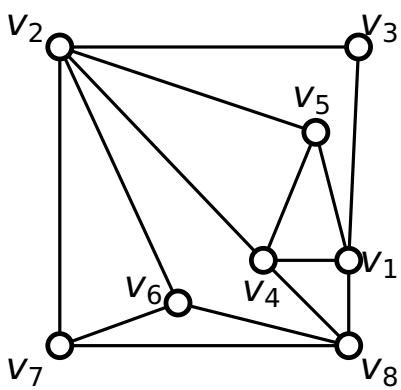
$v_0v_2 \vee v_1v_3$



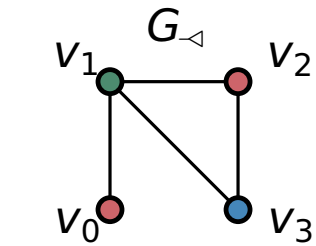
v_1v_3

First Counterexample

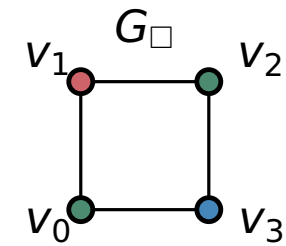
Claim 1. SAT formula is satisfiable ⇔ graph is 3-colorable



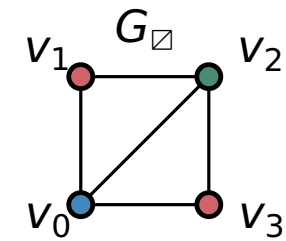
$$(v_3v_4 \vee v_3v_5) \wedge (\neg v_3v_4 \vee \neg v_3v_5) \wedge$$
$$(v_1v_6 \vee v_1v_7) \wedge (\neg v_1v_6 \vee \neg v_1v_7) \wedge$$
$$(v_4v_6 \vee v_4v_7) \wedge (\neg v_4v_6 \vee \neg v_4v_7) \wedge$$
$$(v_5v_6 \vee v_5v_7) \wedge (\neg v_5v_6 \vee \neg v_5v_7) \wedge$$
$$(v_5v_6 \vee v_5v_7) \wedge (\neg v_5v_6 \vee \neg v_5v_7) \wedge$$
$$(v_4v_7 \vee v_5v_7) \wedge (\neg v_4v_7 \vee \neg v_5v_7) \wedge$$
$$(v_1v_2 \vee v_3v_4) \wedge$$
$$(v_1v_2 \vee v_3v_5) \wedge$$
$$(v_1v_2) \wedge$$
$$(v_5v_8)$$



$(v_0v_2 \vee v_0v_3) \wedge$
$(\neg v_0v_2 \vee \neg v_0v_3)$



$v_0v_2 \vee v_1v_3$



v_1v_3

True	False
v_1v_2	v_1v_6
v_1v_7	v_4v_7
v_2v_5	v_5v_6
v_3v_4	v_3v_5
v_4v_6	
v_5v_7	
v_5v_8	

Second Counterexample - Backtracking

Claim 2. Backtracking tree has $O(\log n)$ leaves

Second Counterexample - Backtracking

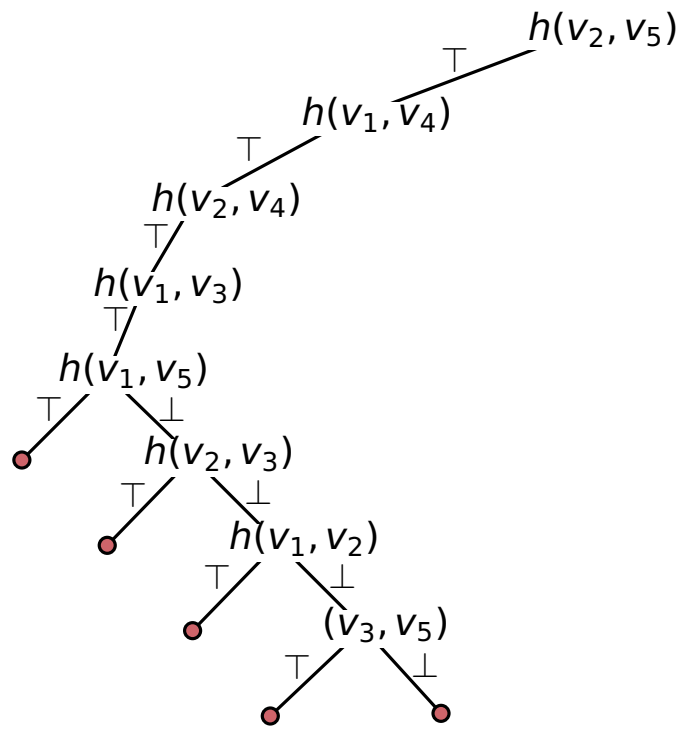
Claim 2. Backtracking tree has $O(\log n)$ leaves

- cycles of length ≥ 5 produce larger clauses
- Unger proposes modified backtracking algorithm

Second Counterexample - Backtracking

Claim 2. Backtracking tree has $O(\log n)$ leaves

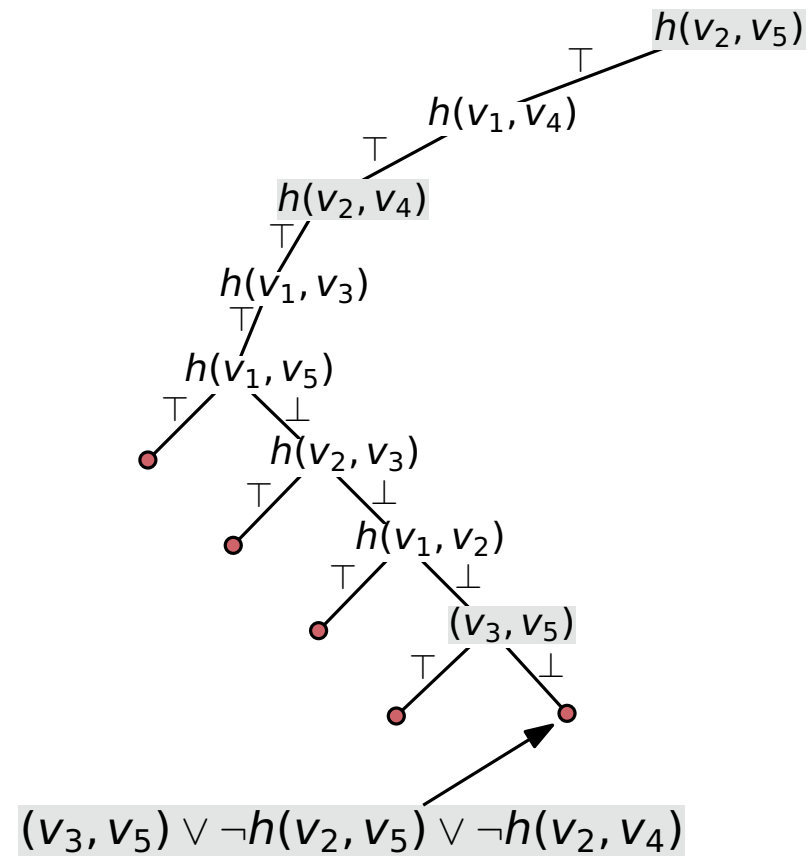
- cycles of length ≥ 5 produce larger clauses
- Unger proposes modified backtracking algorithm
  - jump in tree when all literals in clause are set to false



Second Counterexample - Backtracking

Claim 2. Backtracking tree has $O(\log n)$ leaves

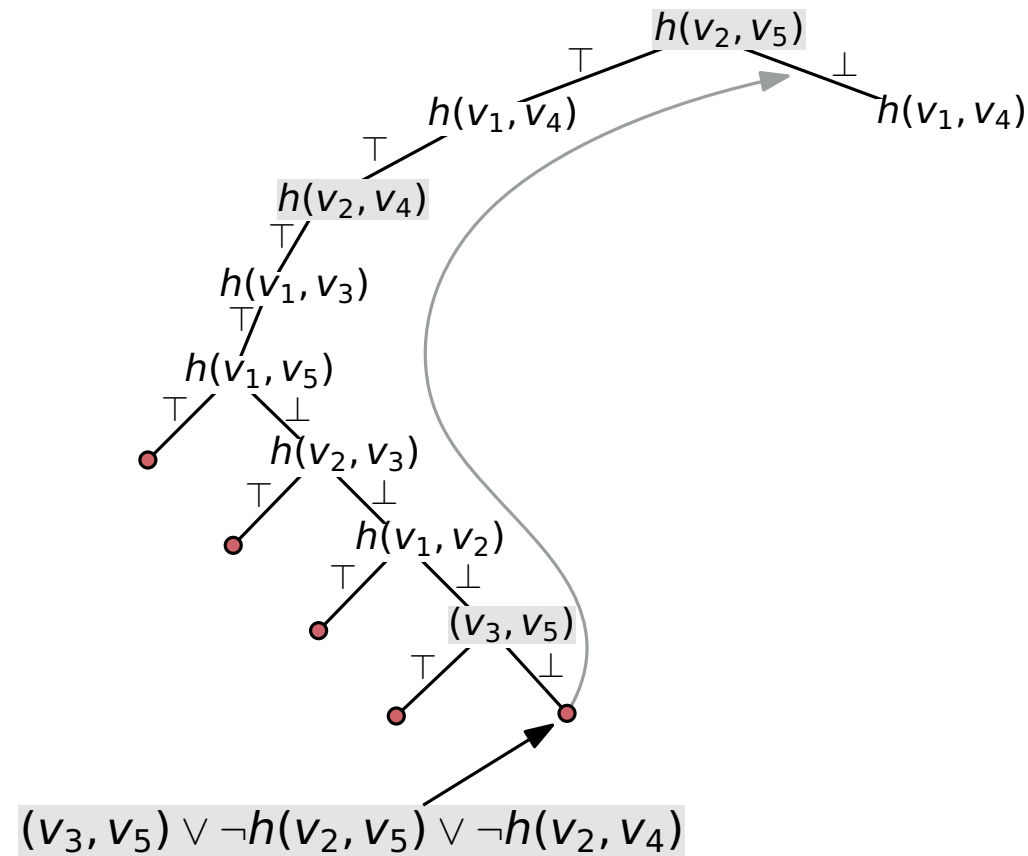
- cycles of length ≥ 5 produce larger clauses
- Unger proposes modified backtracking algorithm
  - jump in tree when all literals in clause are set to false



Second Counterexample - Backtracking

Claim 2. Backtracking tree has $O(\log n)$ leaves

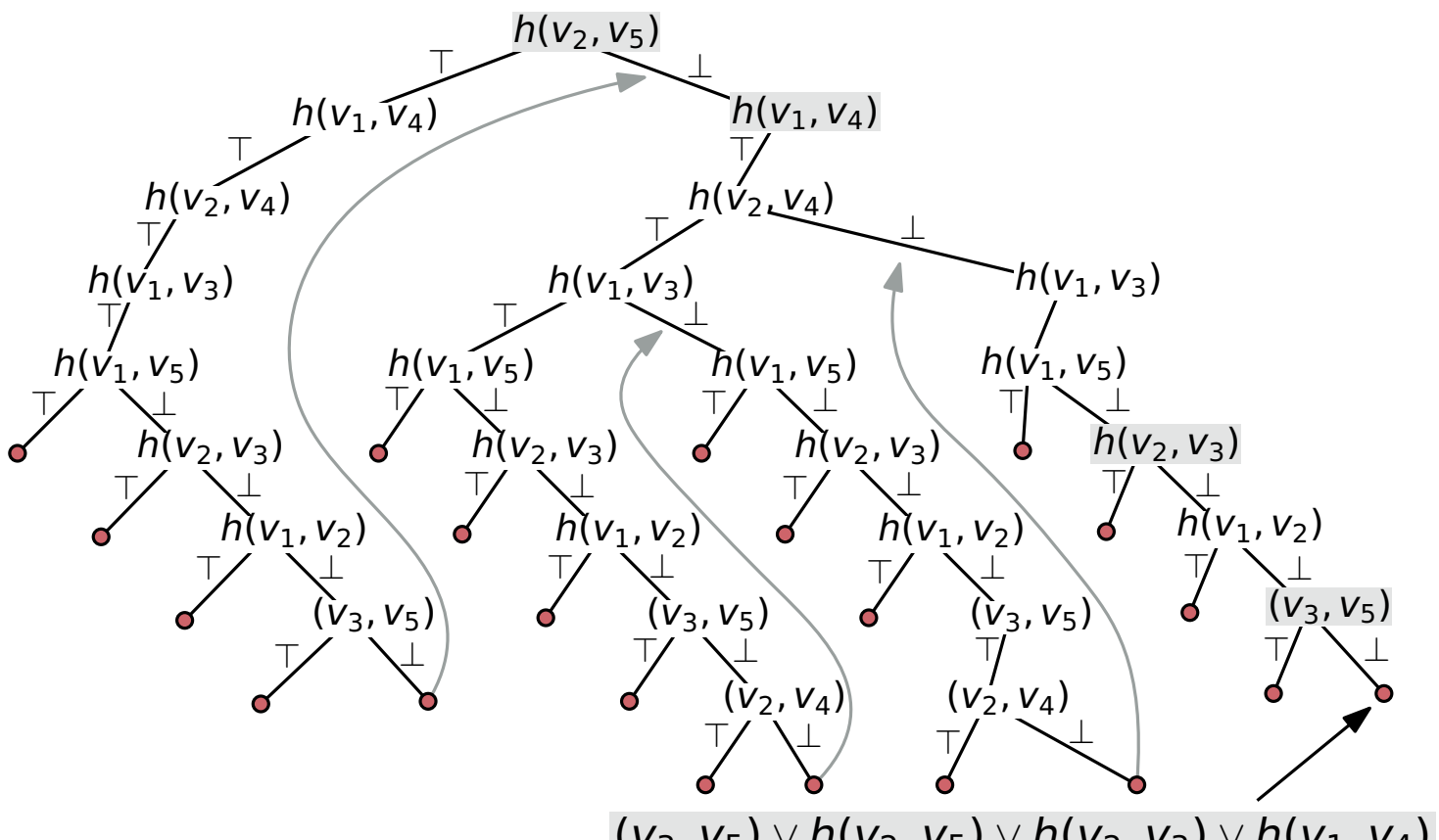
- cycles of length ≥ 5 produce larger clauses
- Unger proposes modified backtracking algorithm
  - jump in tree when all literals in clause are set to false



Second Counterexample - Backtracking

Claim 2. Backtracking tree has $O(\log n)$ leaves

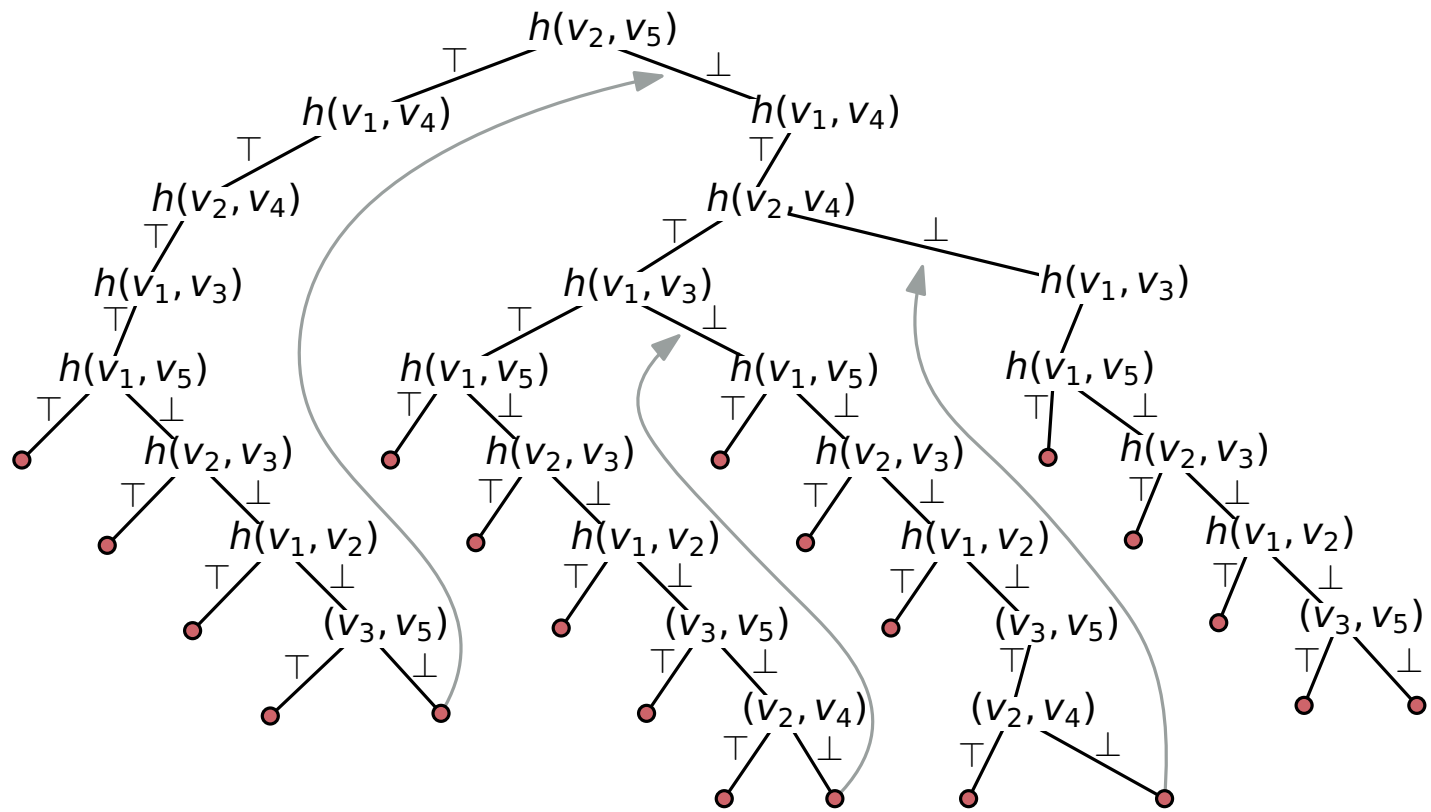
- cycles of length ≥ 5 produce larger clauses
- Unger proposes modified backtracking algorithm
  - jump in tree when all literals in clause are set to false
  - if no such variable exists, algorithm reports not 3-colorable



Second Counterexample - Backtracking

Claim 2. Backtracking tree has $O(\log n)$ leaves

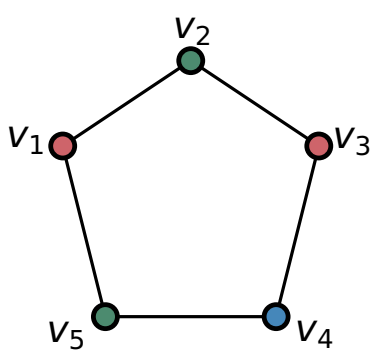
- cycles of length ≥ 5 produce larger clauses
- Unger proposes modified backtracking algorithm
  - jump in tree when all literals in clause are set to false
  - if no such variable exists, algorithm reports not 3-colorable
- order in which variables are assigned is not specified

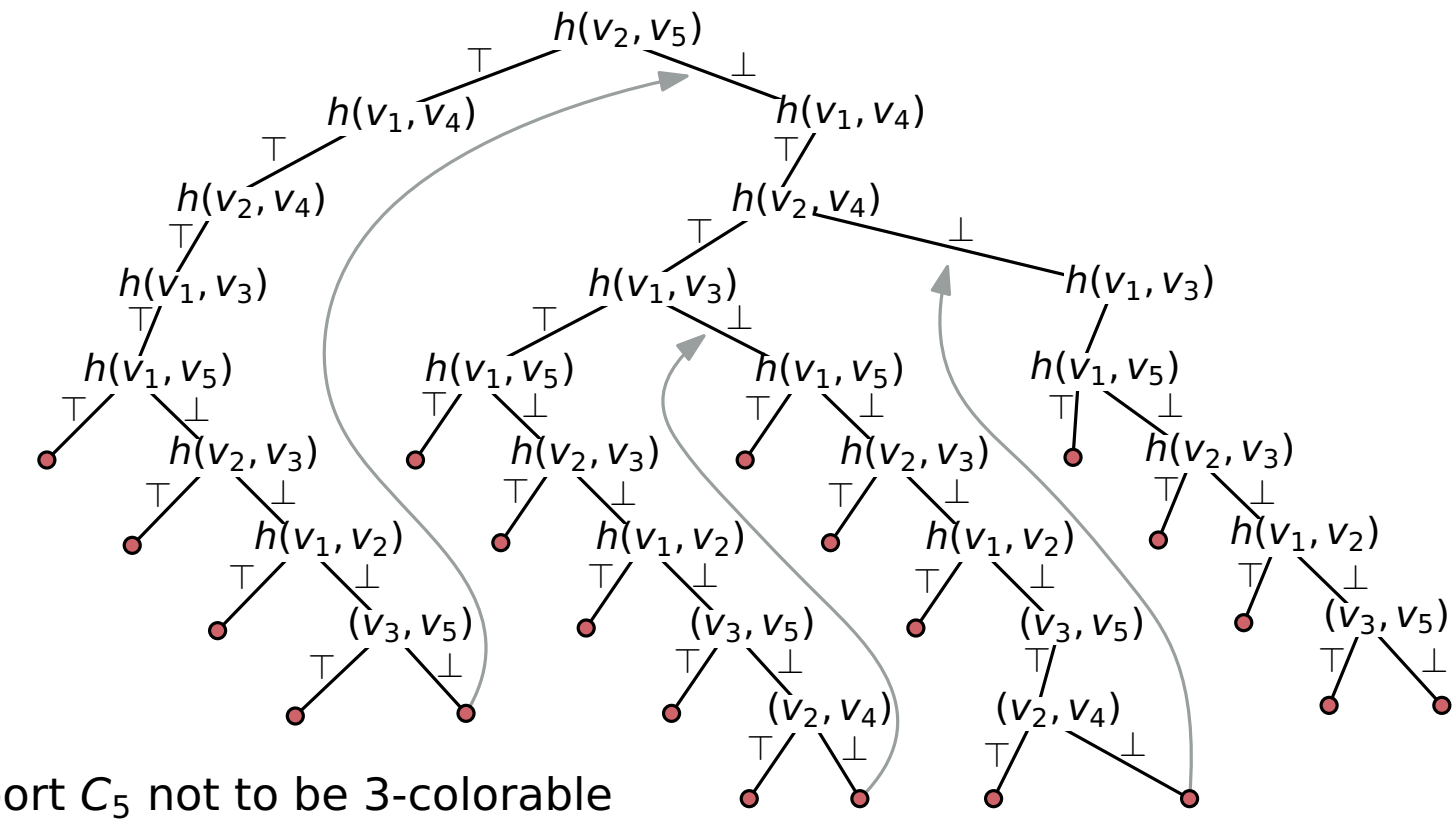


Second Counterexample - Backtracking

Claim 2. Backtracking tree has $O(\log n)$ leaves

- cycles of length ≥ 5 produce larger clauses
- Unger proposes modified backtracking algorithm
  - jump in tree when all literals in clause are set to false
  - if no such variable exists, algorithm reports not 3-colorable
- order in which variables are assigned is not specified





- backtracking might report C_5 not to be 3-colorable

Evaluation

Goal: show how well Unger’s algorithms hold up in practice

Evaluation

Goal: show how well Unger’s algorithms hold up in practice

Test data: 2196 random circle graphs with up to 750 vertices

- 3-colorable
- Connected
- No cliques of size 4 or larger

Evaluation

Goal: show how well Unger’s algorithms hold up in practice

Test data: 2196 random circle graphs with up to 750 vertices

- 3-colorable
- Connected
- No cliques of size 4 or larger

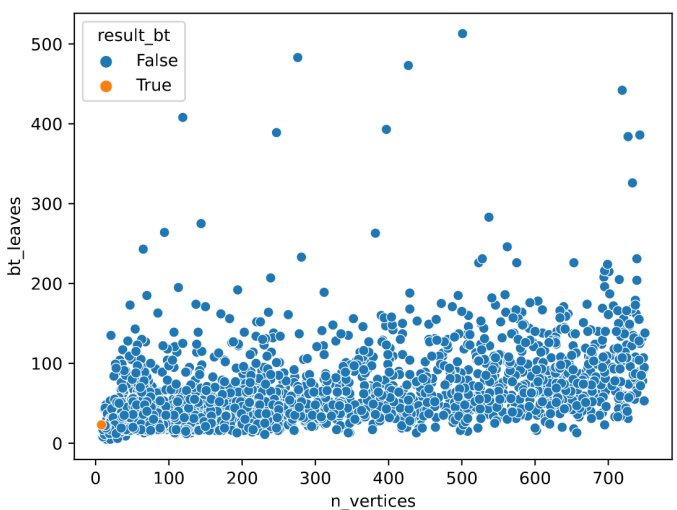


Fig. 1. Number of leaves in Unger’s backtracking tree.

Evaluation

Goal: show how well Unger's algorithms hold up in practice

Test data: 2196 random circle graphs with up to 750 vertices

- 3-colorable
- Connected
- No cliques of size 4 or larger

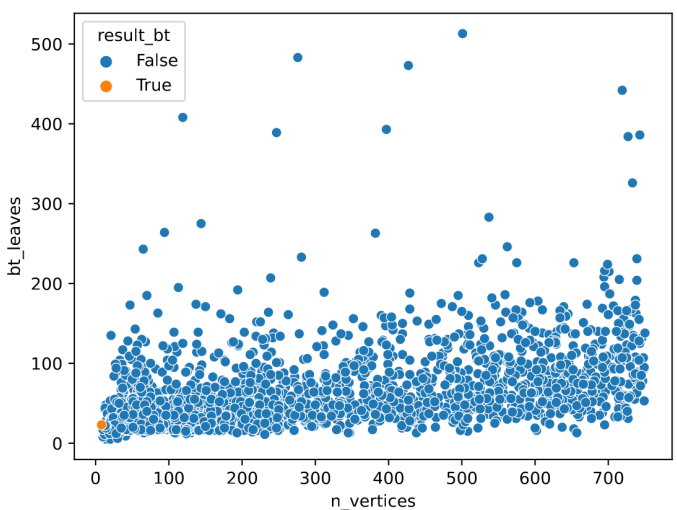


Fig. 1. Number of leaves in Unger's backtracking tree.

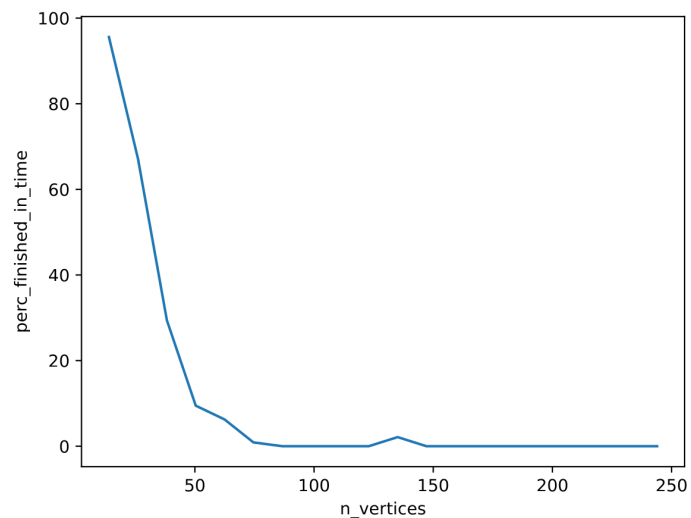


Fig. 2. Instances solved by regular backtracking within one-hour time limit.

Evaluation

Goal: show how well Unger's algorithms hold up in practice

Test data: 2196 random circle graphs with up to 750 vertices

- 3-colorable
- Connected
- No cliques of size 4 or larger

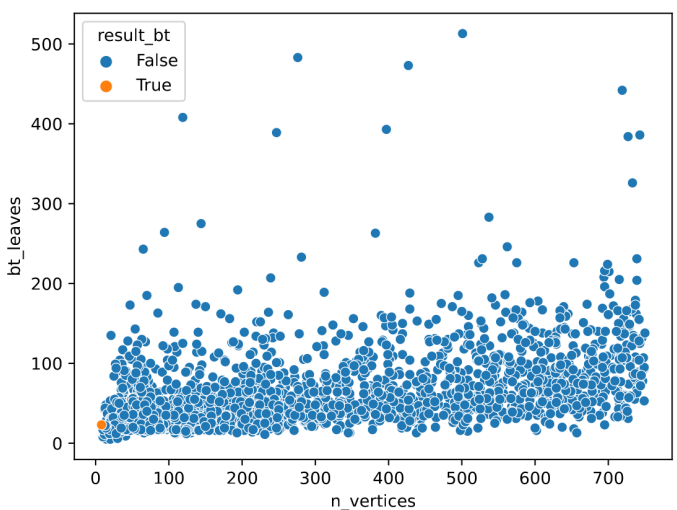


Fig. 1. Number of leaves in Unger's backtracking tree.

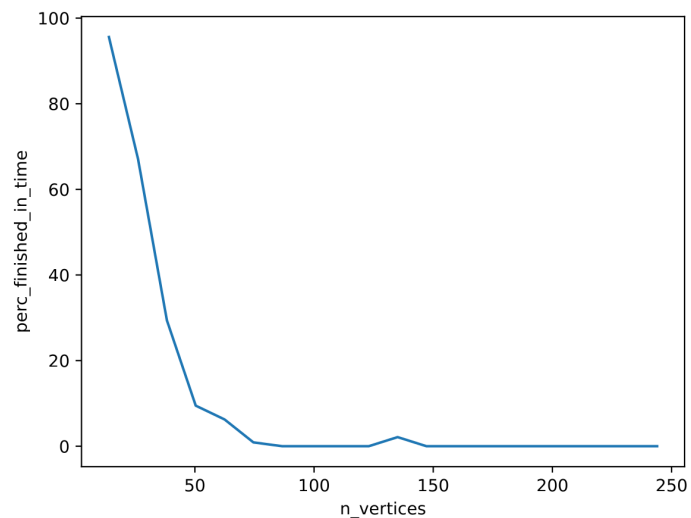


Fig. 2. Instances solved by regular backtracking within one-hour time limit.

- #backtracking leaves likely not consistent with claimed $O(\log n)$ upper bound

Evaluation

Goal: show how well Unger's algorithms hold up in practice

Test data: 2196 random circle graphs with up to 750 vertices

- 3-colorable
- Connected
- No cliques of size 4 or larger

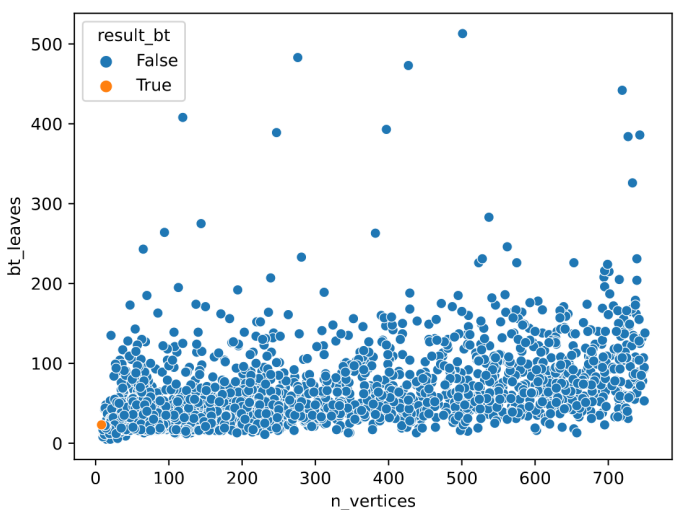


Fig. 1. Number of leaves in Unger's backtracking tree.

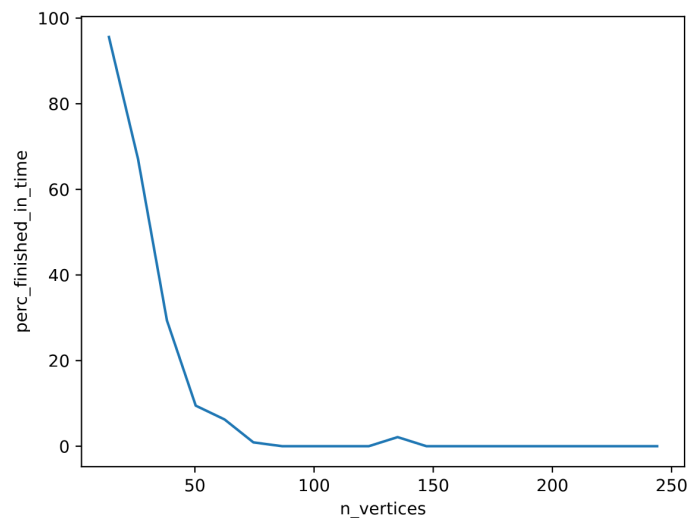


Fig. 2. Instances solved by regular backtracking within one-hour time limit.

- #backtracking leaves likely not consistent with claimed $O(\log n)$ upper bound
- no solution for all but one instance

Evaluation

Goal: show how well Unger's algorithms hold up in practice

Test data: 2196 random circle graphs with up to 750 vertices

- 3-colorable
- Connected
- No cliques of size 4 or larger

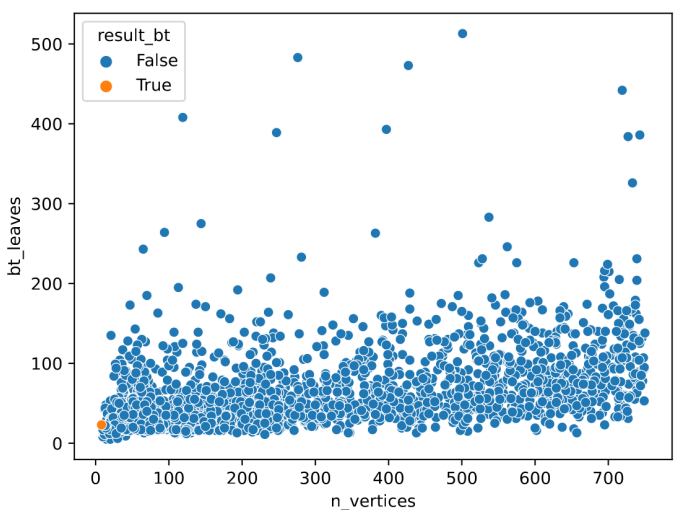


Fig. 1. Number of leaves in Unger's backtracking tree.

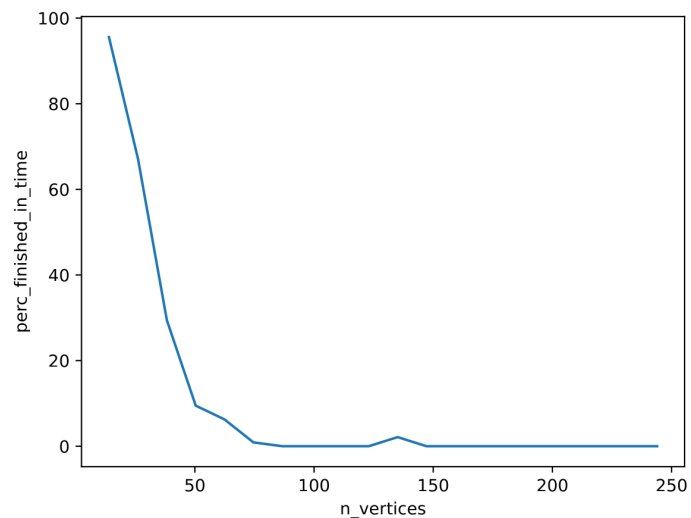


Fig. 2. Instances solved by regular backtracking within one-hour time limit.

- #backtracking leaves likely not consistent with claimed $O(\log n)$ upper bound
- no solution for all but one instance
- regular backtracking fails to compute solution for $|V| \geq 50$ within one hour

Evaluation

Goal: show how well Unger's algorithms hold up in practice

Test data: 2196 random circle graphs with up to 750 vertices

- 3-colorable
- Connected
- No cliques of size 4 or larger

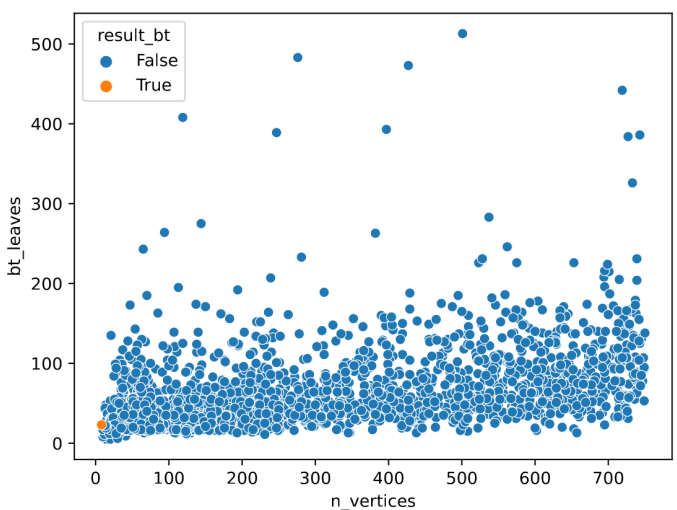


Fig. 1. Number of leaves in Unger's backtracking tree.

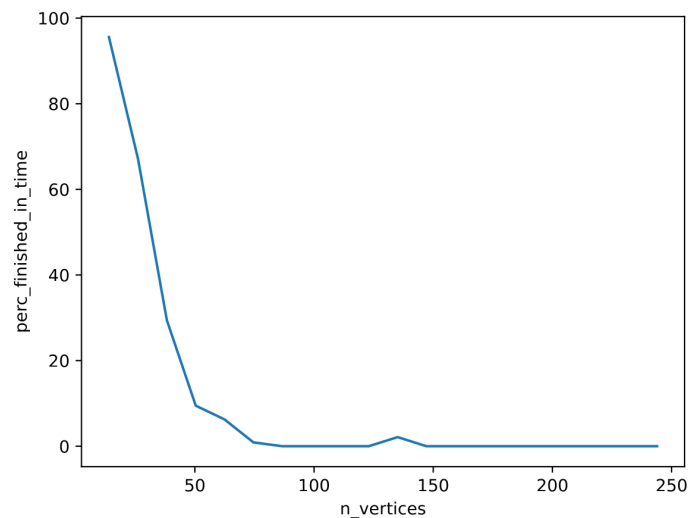


Fig. 2. Instances solved by regular backtracking within one-hour time limit.

- #backtracking leaves likely not consistent with claimed $O(\log n)$ upper bound
- no solution for all but one instance
- regular backtracking fails to compute solution for $|V| \geq 50$ within one hour
- less than 20% of SAT solutions gave valid colorings

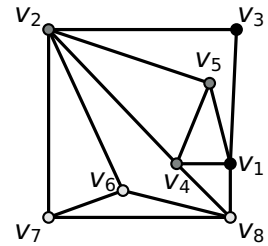
Conclusion

Results

Conclusion

Results

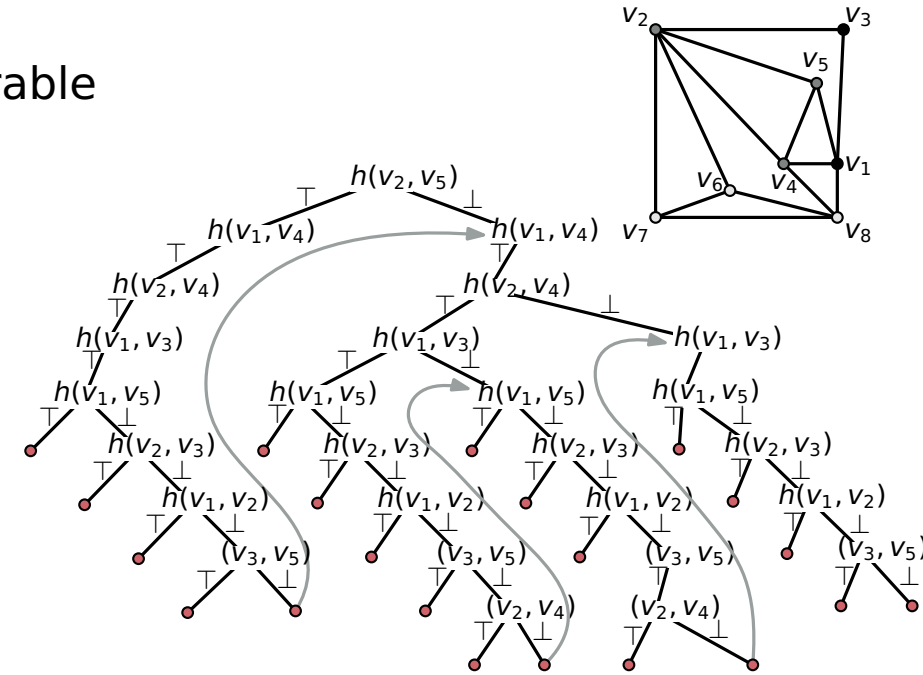
- satisfiable SAT instance not equivalent to 3-colorable



Conclusion

Results

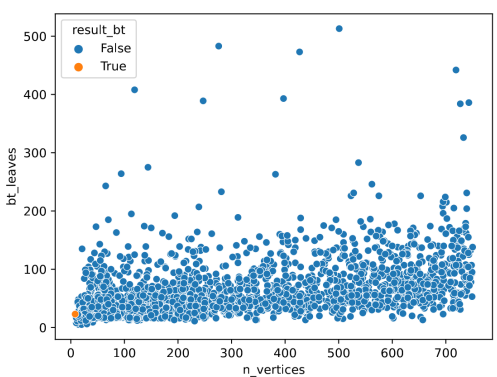
- satisfiable SAT instance not equivalent to 3-colorable
- backtracking algorithm might give wrong result

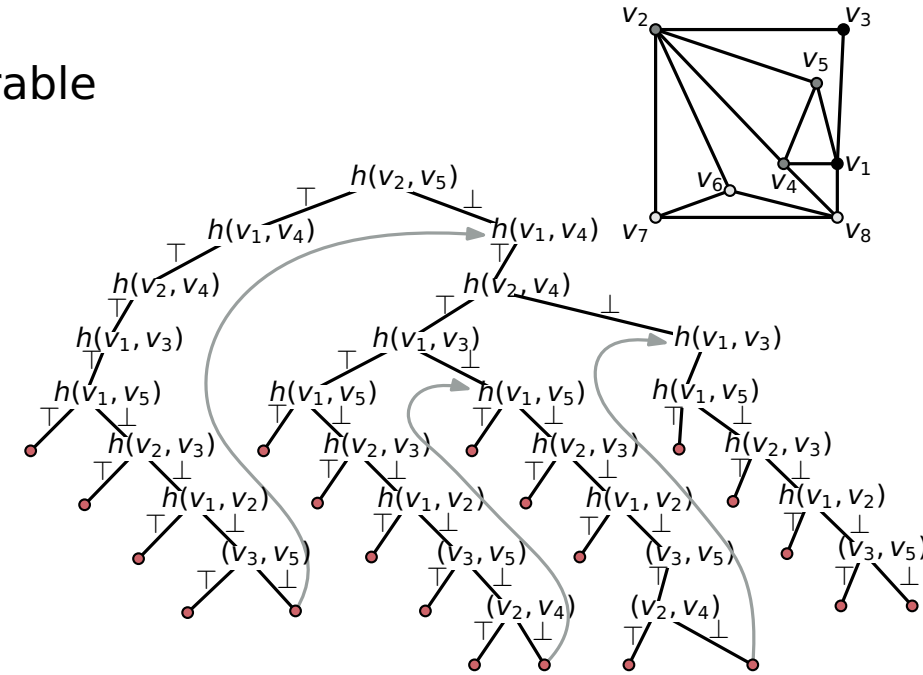


Conclusion

Results

- satisfiable SAT instance not equivalent to 3-colorable
- backtracking algorithm might give wrong result
- evaluation gave results not fitting the claimed upper bound for #leaves

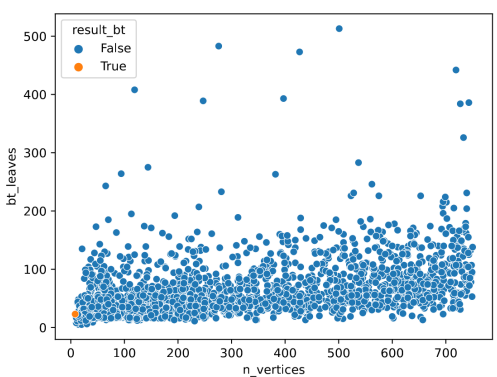


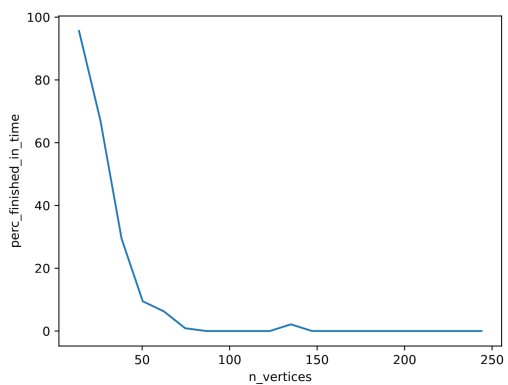


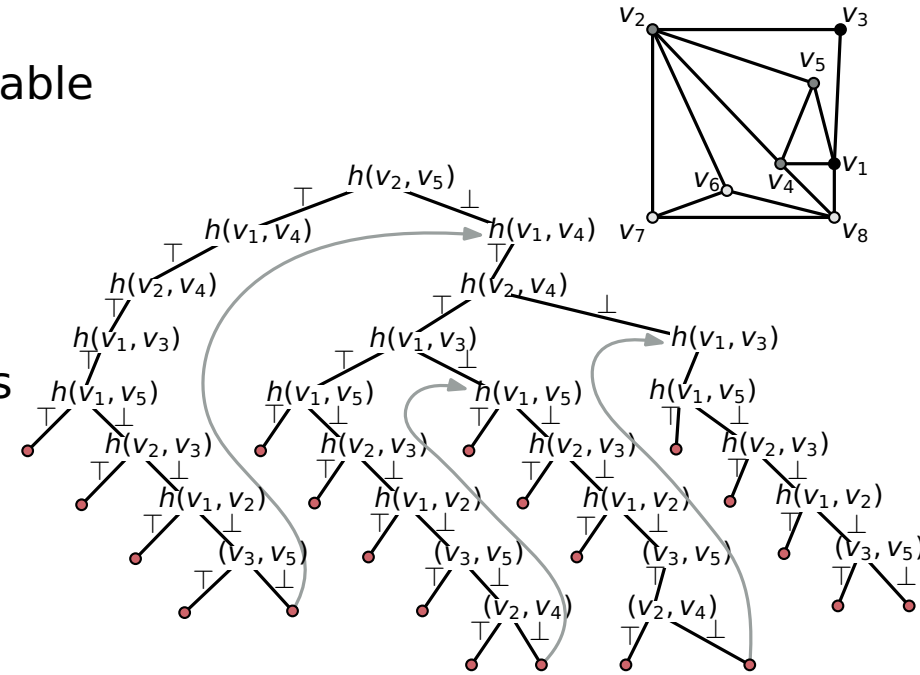
Conclusion

Results

- satisfiable SAT instance not equivalent to 3-colorable
- backtracking algorithm might give wrong result
- evaluation gave results not fitting the claimed upper bound for #leaves
- regular backtracking impractical on circle graphs



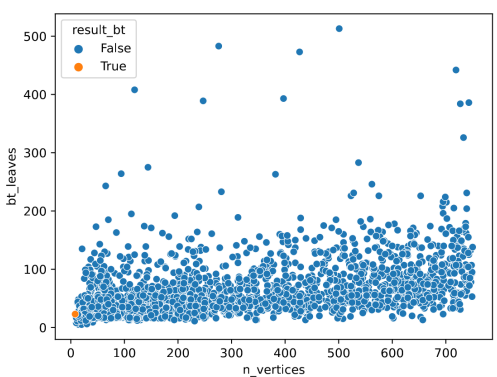


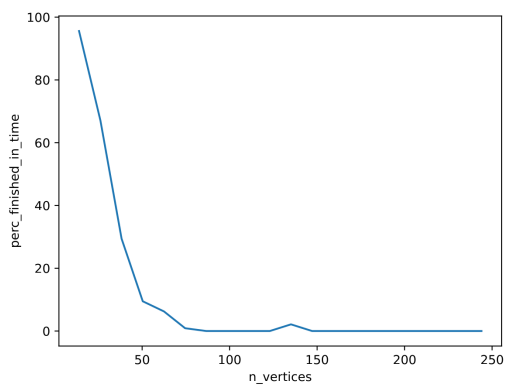


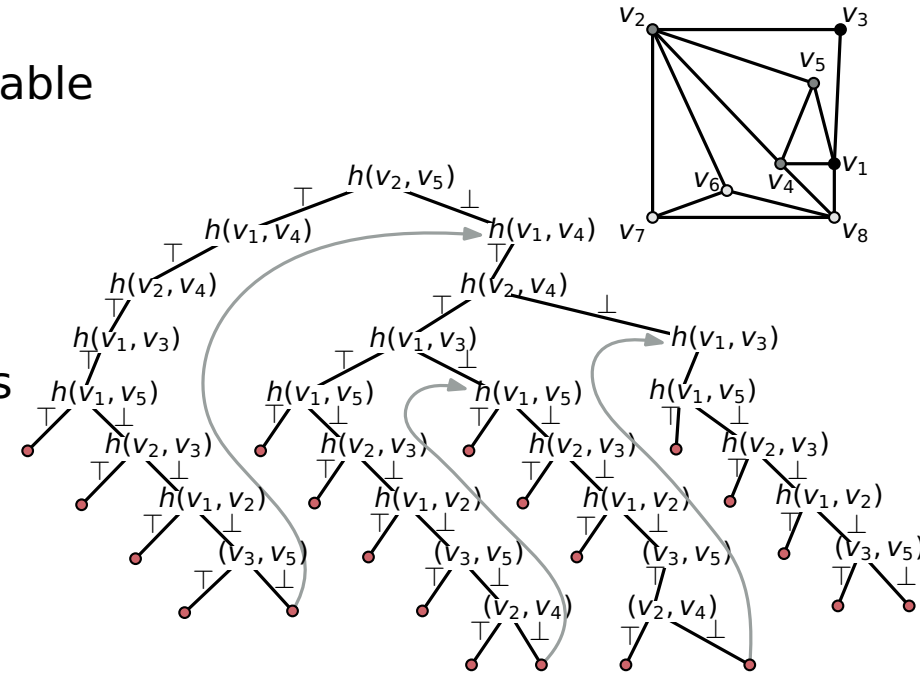
Conclusion

Results

- satisfiable SAT instance not equivalent to 3-colorable
- backtracking algorithm might give wrong result
- evaluation gave results not fitting the claimed upper bound for #leaves
- regular backtracking impractical on circle graphs





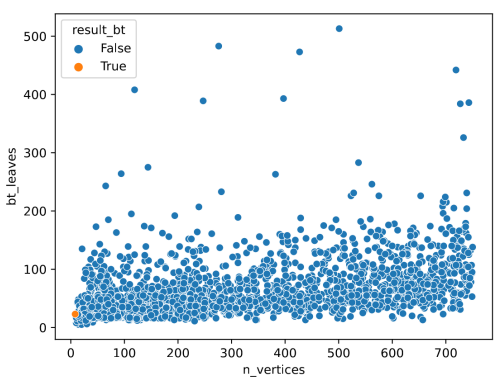


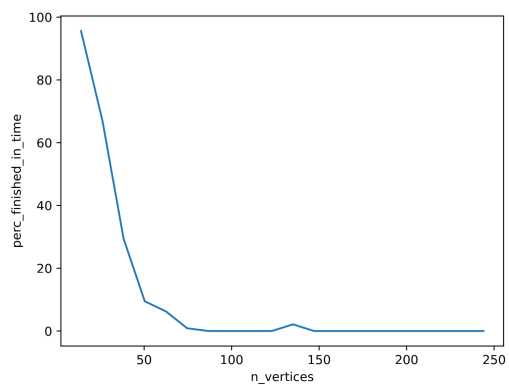
- colorings constructed from Sat instance often not valid

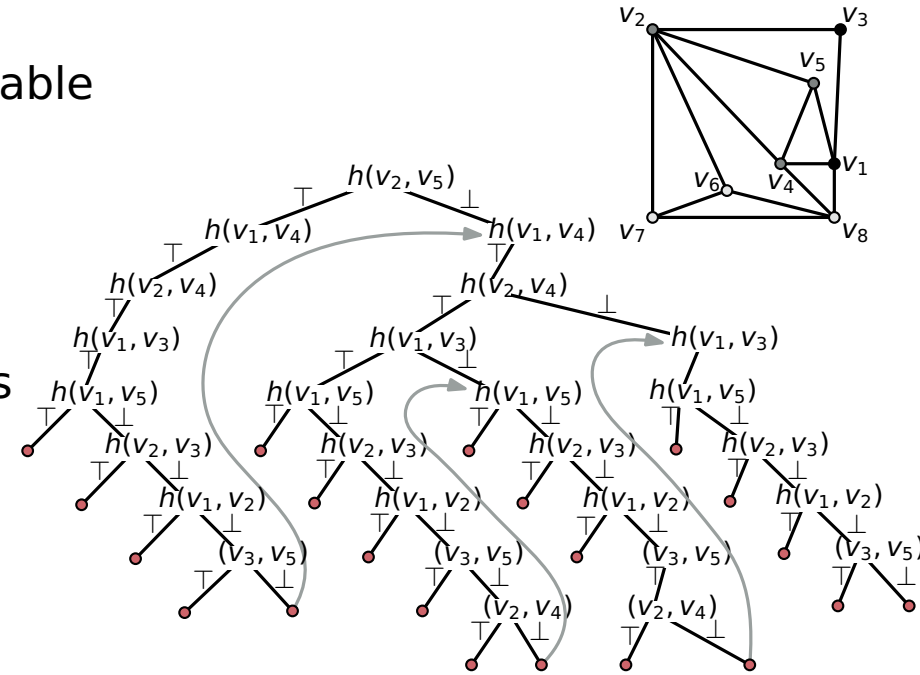
Conclusion

Results

- satisfiable SAT instance not equivalent to 3-colorable
- backtracking algorithm might give wrong result
- evaluation gave results not fitting the claimed upper bound for #leaves
- regular backtracking impractical on circle graphs







- colorings constructed from Sat instance often not valid

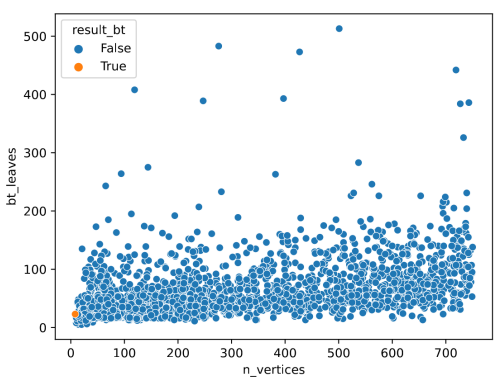
Open Problems

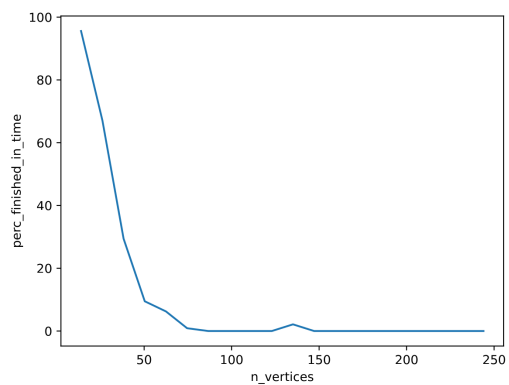
- 3-coloring circle graphs

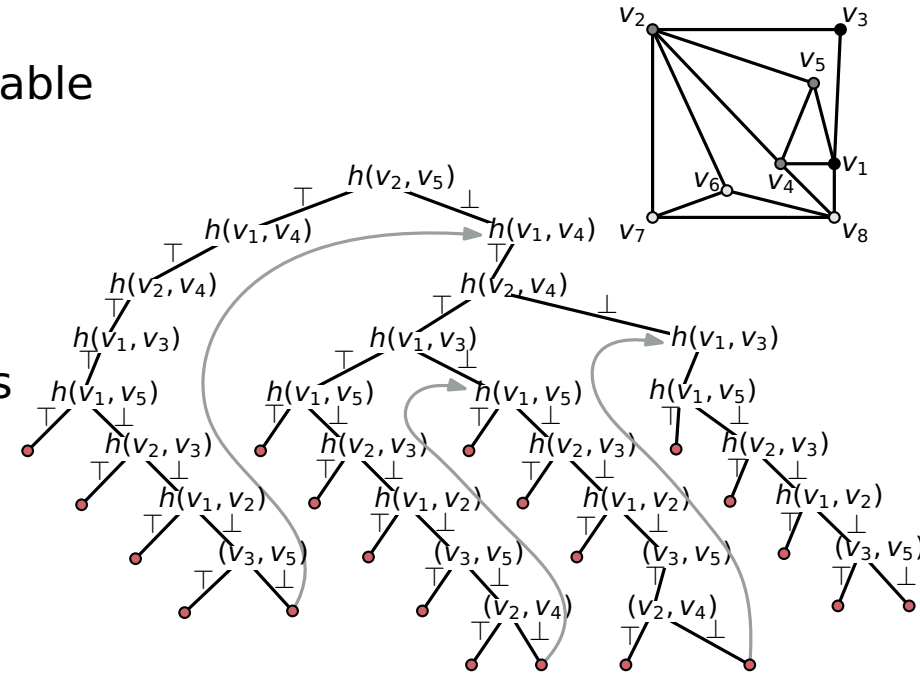
Conclusion

Results

- satisfiable SAT instance not equivalent to 3-colorable
- backtracking algorithm might give wrong result
- evaluation gave results not fitting the claimed upper bound for #leaves
- regular backtracking impractical on circle graphs







- colorings constructed from Sat instance often not valid

Open Problems

- 3-coloring circle graphs
- complexity of 3-coloring circle graphs with no induced cycles C_k where $k > 4$