The Parameterized Complexity of Extending Stack Layouts

Thomas Depian, Simon D. Fink,
Robert Ganian, Martin Nöllenburg

18. – 20. September · GD 2024

TU WIEN
ac ALGORITHMS AND COMPLEXITY GROUP

Stack Layouts

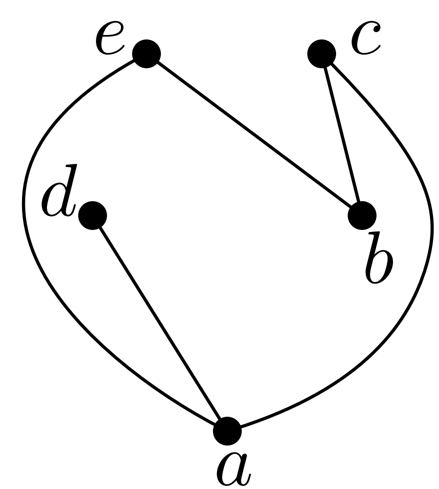
Stack Layout

Given:

Integer $\ell > 0$

Graph G

$\ell = 2$



Stack Layouts

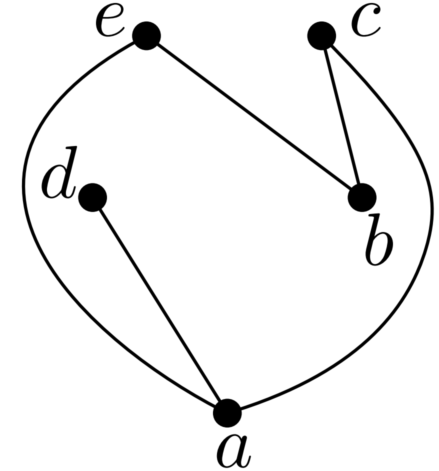
Stack Layout

Given:

Integer $\ell > 0$

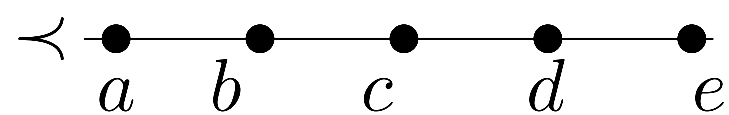
Graph G

$\ell = 2$



Want:

Linear order $\prec$ of vertices $V(G)$



Stack Layouts

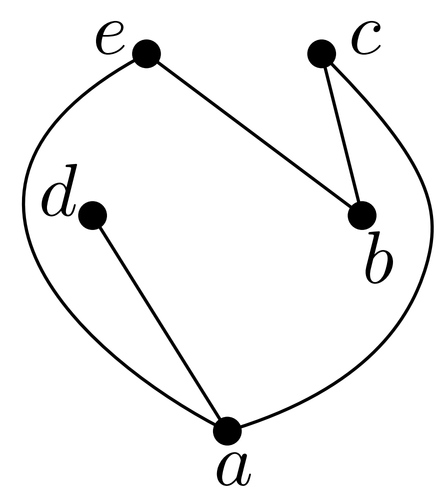
STACK LAYOUT

Given:

Integer $\ell > 0$

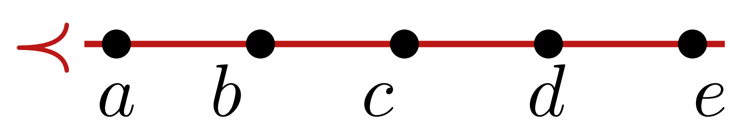
Graph G

$\ell = 2$



Want:

Linear order $\prec$ of vertices $V(G)$



Stack Layouts

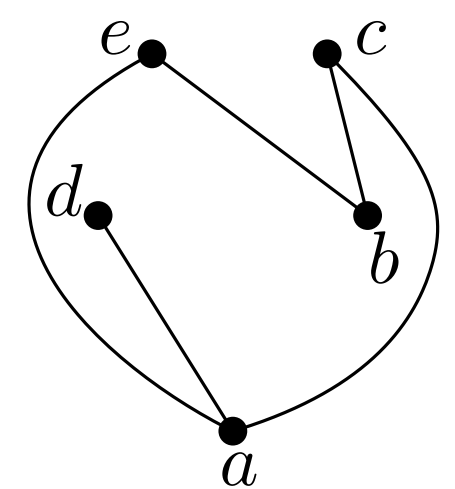
Stack Layout

Given:

Integer $\ell > 0$

Graph G

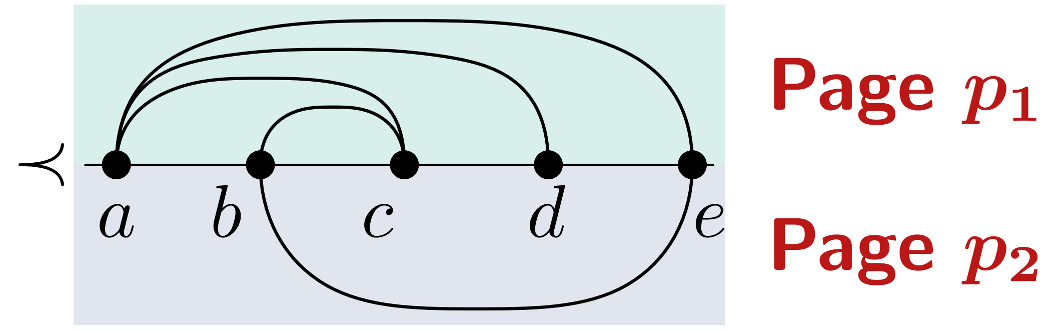
$\ell = 2$



Want:

Linear order $\prec$ of vertices $V(G)$

Page assignment $\sigma\colon E(G) \to [\ell]$



Stack Layouts

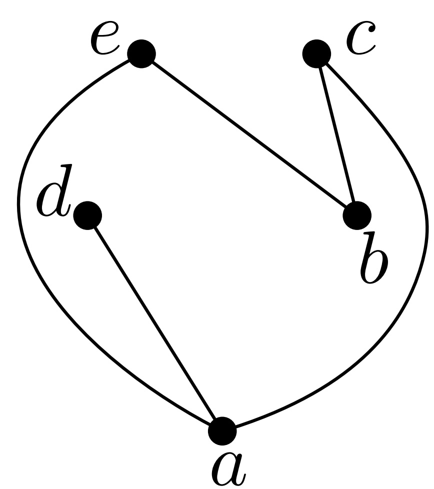
Stack Layout

Given:

Integer $\ell > 0$

Graph G

$\ell = 2$

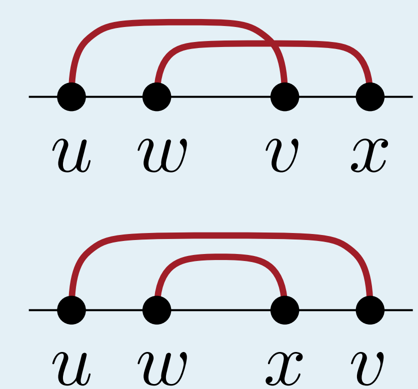


Want:

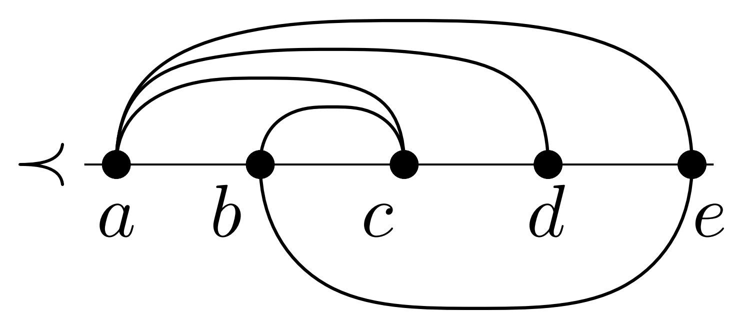
Linear order $\prec$ of vertices $V(G)$

Page assignment $\sigma\colon E(G) \to [\ell]$

Such that no two edges on the same page cross



endpoints of no two edges on the same page alternate



Stack Layouts

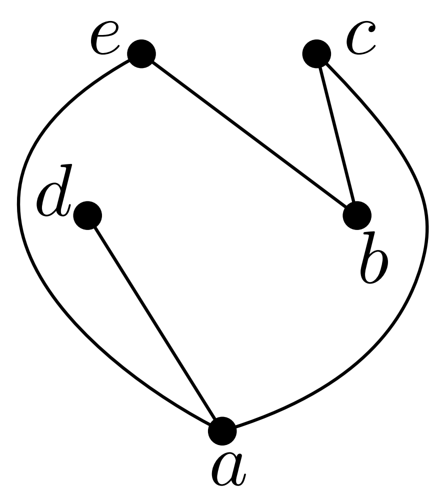
Stack Layout

Given:

Integer $\ell > 0$

Graph G

$\ell = 2$



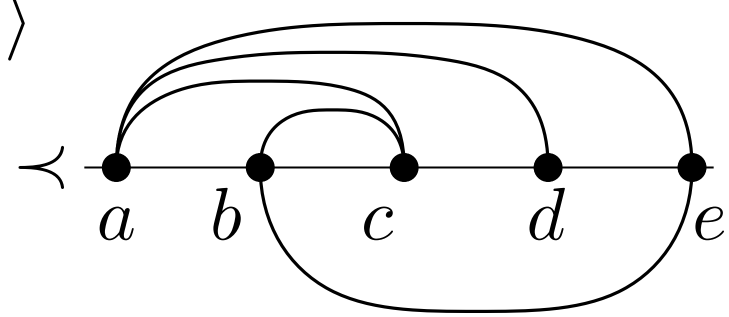
Want:

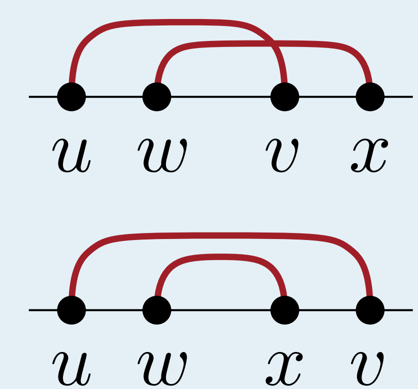
Linear order $\prec$ of vertices $V(G)$

Page assignment $\sigma\colon E(G) \to [\ell]$

Such that no two edges on the same page cross

ℓ-page stack layout $\langle \prec_G, \sigma_G \rangle$





endpoints of no two edges on the same page alternate

Stack Layouts

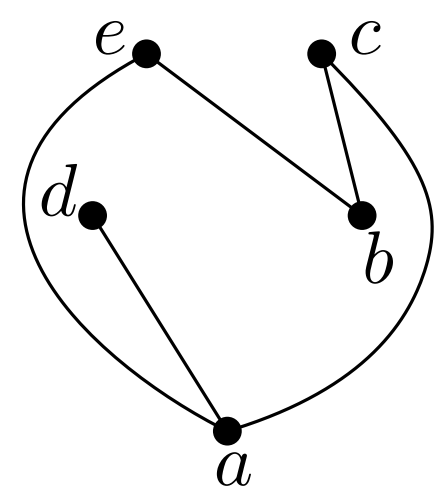
Stack Layout

Given:

Integer $\ell > 0$

Graph G

$\ell = 2$



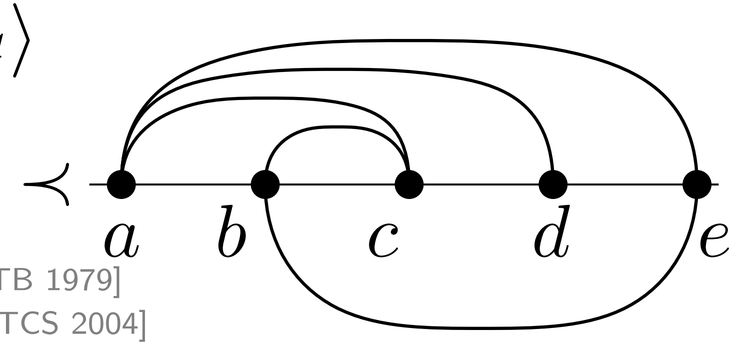
Want:

Linear order $\prec$ of vertices $V(G)$

Page assignment $\sigma\colon E(G) \to [\ell]$

Such that no two edges on the same page cross

ℓ-page stack layout $\langle \prec_G, \sigma_G \rangle$



Well-studied field [Bernhard and Kainen, JCTB 1979] [Dujmović and Wood, DMTCS 2004]

Stack Layouts

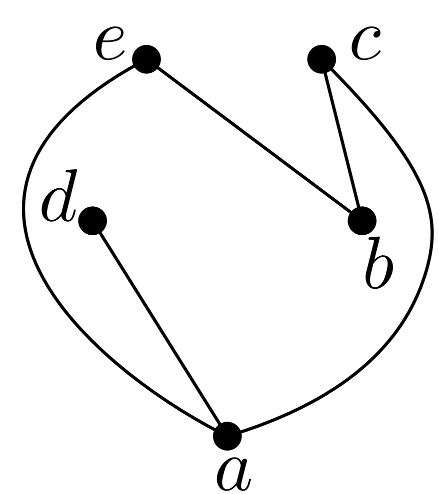
Stack Layout

Given:

Integer $\ell > 0$

Graph G

$\ell = 2$



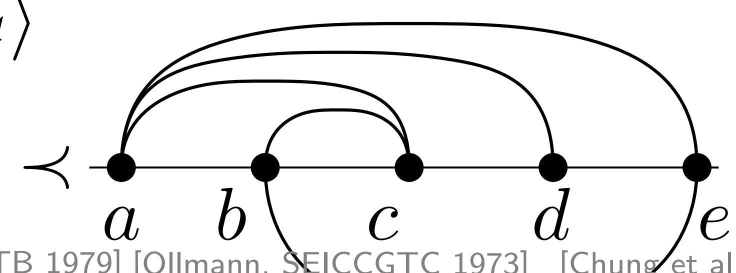
Want:

Linear order $\prec$ of vertices $V(G)$

Page assignment $\sigma\colon E(G) \to [\ell]$

Such that no two edges on the same page cross

ℓ-page stack layout $\langle \prec_G, \sigma_G \rangle$



[Yannakakis, JCSS 1998] [Bekos et al., JOCG 2020]

Well-studied field [Bernhard and Kainen, JCTB 1979] [Ollmann, SEICCGTC 1973] [Chung et al., JADM 1987] [Bhore et al., JGAA 2020] [Liu et al., TCS 2021] [Dujmović and Wood, DMTCS 2004] [Bilski, IEEE Proc. E 1992] [Unger, STACS 1988] [Ganian et al., ICALP 2024]

Stack Layouts

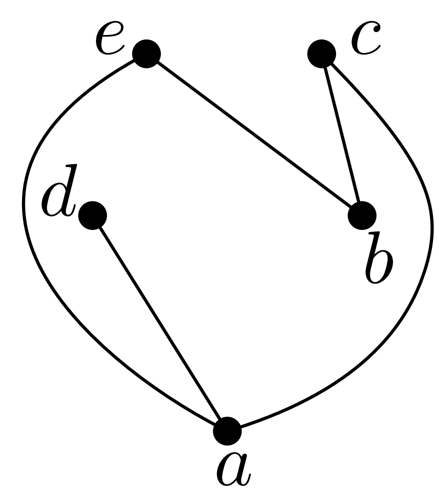
Stack Layout

Given:

Integer $\ell > 0$

Graph G

$\ell = 2$



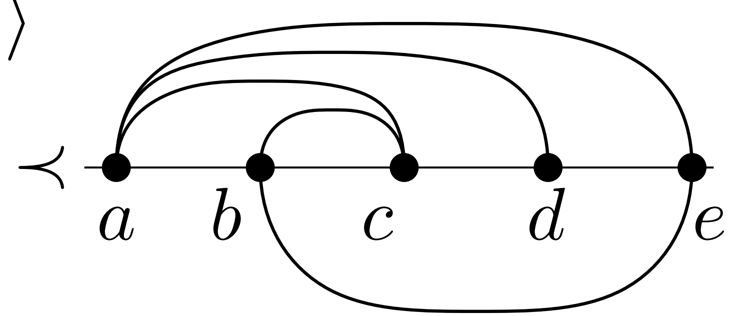
Want:

Linear order $\prec$ of vertices $V(G)$

Page assignment $\sigma\colon E(G) \to [\ell]$

Such that no two edges on the same page cross

ℓ-page stack layout $\langle \prec_G, \sigma_G \rangle$



Well-studied field

Known:

NP-complete ($\ell = 2$)

Stack Layouts

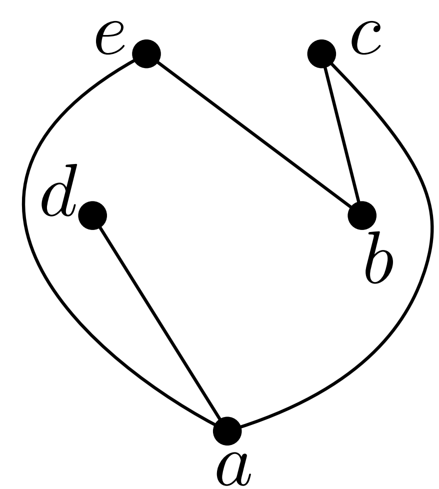
STACK LAYOUT

Given:

Integer $\ell > 0$

Graph G

$\ell = 2$



Known:

NP-complete ($\ell = 2$)

(even if $\prec$ is given & $\ell = 4$)

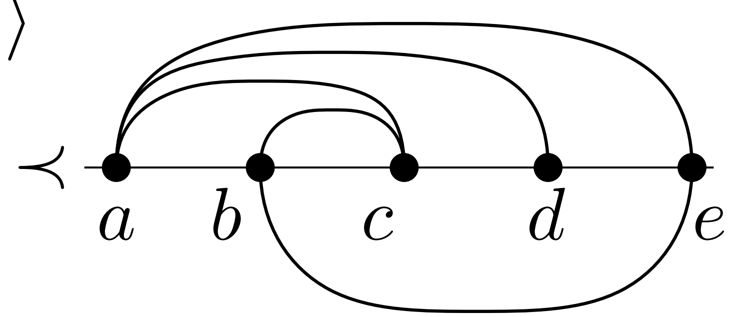
Want:

Linear order $\prec$ of vertices $V(G)$

Page assignment $\sigma\colon E(G) \to [\ell]$

Such that no two edges on the same page cross

ℓ-page stack layout $\langle \prec_G, \sigma_G \rangle$



Well-studied field

Stack Layouts

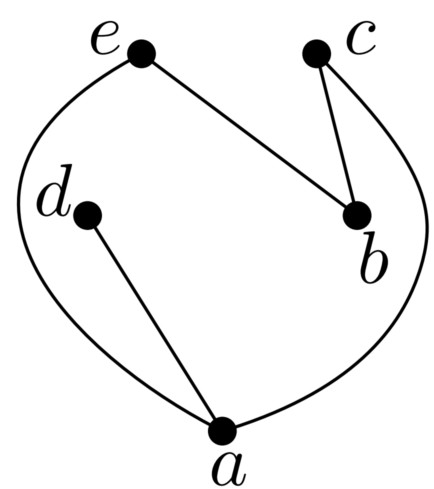
STACK LAYOUT

Given:

Integer $\ell > 0$

Graph G

$\ell = 2$



Known:

NP-complete ($\ell = 2$)

(even if $\prec$ is given & $\ell = 4$)

FPT in vcn of G

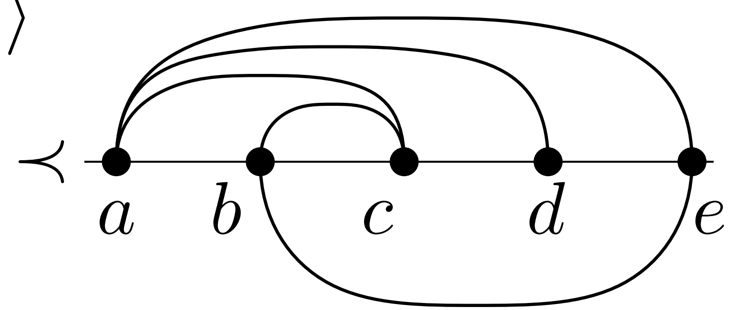
Want:

Linear order $\prec$ of vertices $V(G)$

Page assignment $\sigma \colon E(G) \to [\ell]$

Such that no two edges on the same page cross

ℓ-page stack layout $\langle \prec_G, \sigma_G \rangle$



Well-studied field

Stack Layout Extension

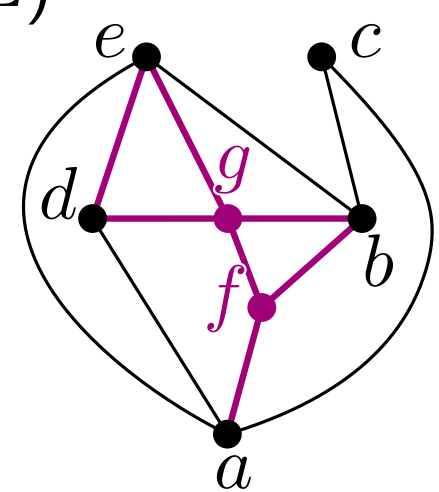
STACK LAYOUT EXTENSION (SLE)

Given:

Integer $\ell > 0$

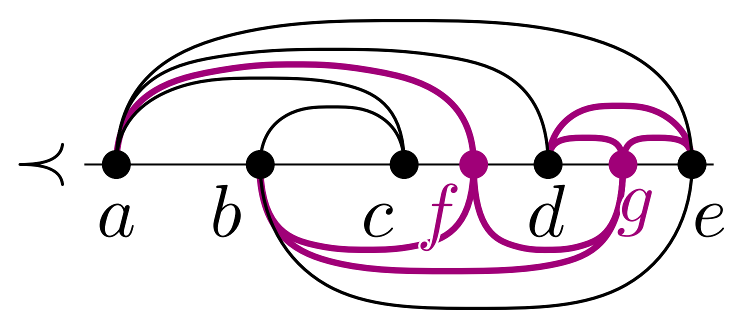
Graph G

$\ell = 2$



Want:

ℓ-page stack layout $\langle \prec_G, \sigma_G \rangle$ of G



Stack Layout Extension

STACK LAYOUT EXTENSION (SLE)

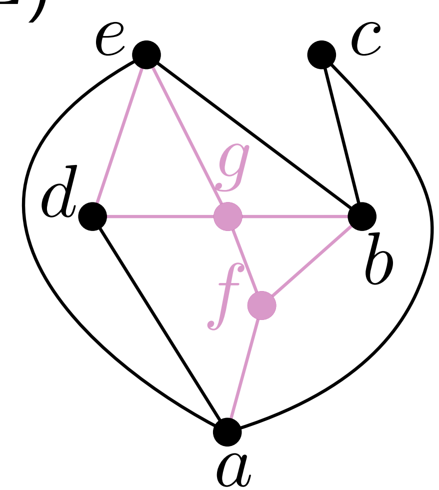
Given:

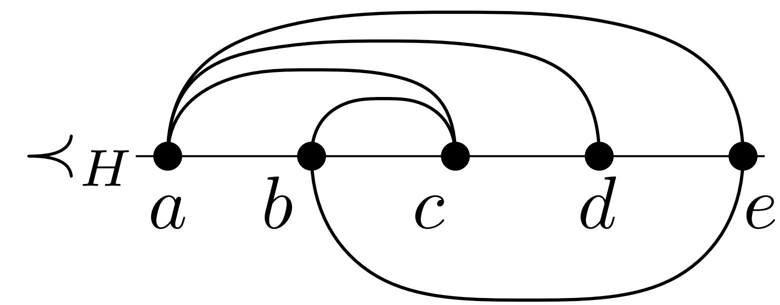
Integer $\ell > 0$

Graph G

ℓ-page stack layout $\langle \prec_H, \sigma_H \rangle$
of subgraph $H \subseteq G$

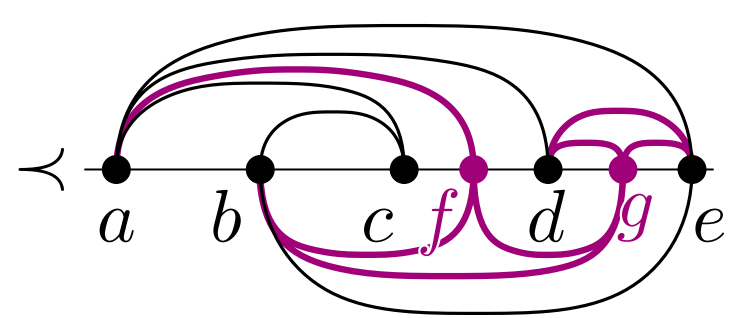
$\ell = 2$





Want:

ℓ-page stack layout $\langle \prec_G, \sigma_G \rangle$ of G



Stack Layout Extension

Stack Layout Extension (SLE)

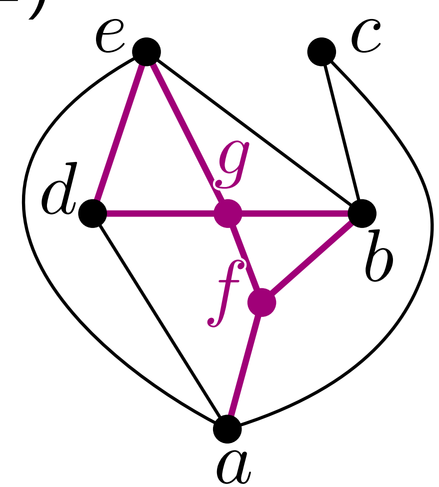
Given:

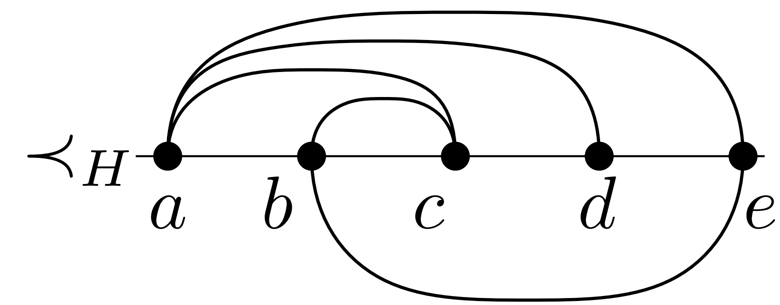
Integer $\ell > 0$

Graph G

ℓ-page stack layout $\langle \prec_H, \sigma_H \rangle$
of subgraph $H \subseteq G$

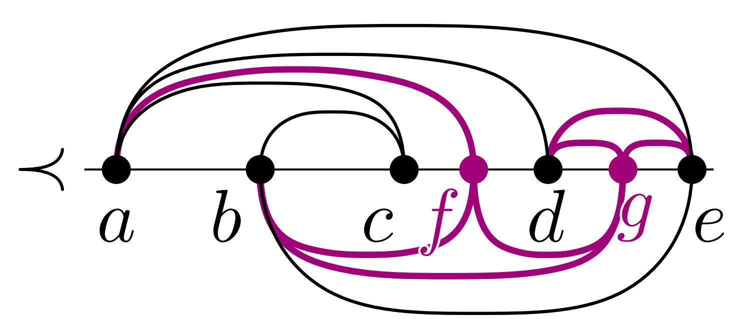
$\ell = 2$





Want:

ℓ-page stack layout $\langle \prec_G, \sigma_G \rangle$ of G that **extends** $\langle \prec_H, \sigma_H \rangle$



Stack Layout Extension

STACK LAYOUT EXTENSION (SLE)

Given:

Integer $\ell > 0$

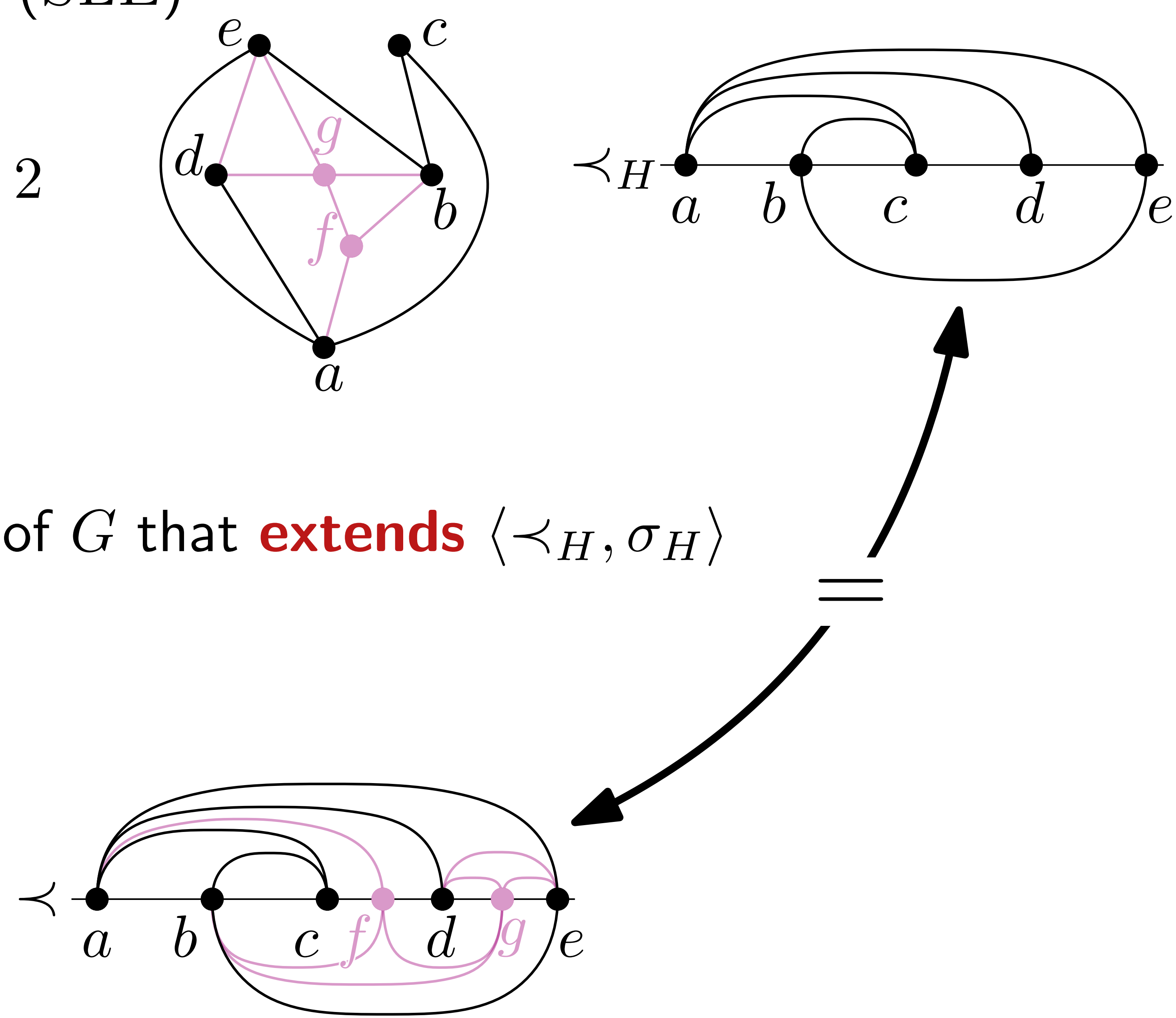
Graph G

ℓ-page stack layout $\langle \prec_H, \sigma_H \rangle$
of subgraph $H \subseteq G$

Want:

ℓ-page stack layout $\langle \prec_G, \sigma_G \rangle$ of G that **extends** $\langle \prec_H, \sigma_H \rangle$

$\ell = 2$



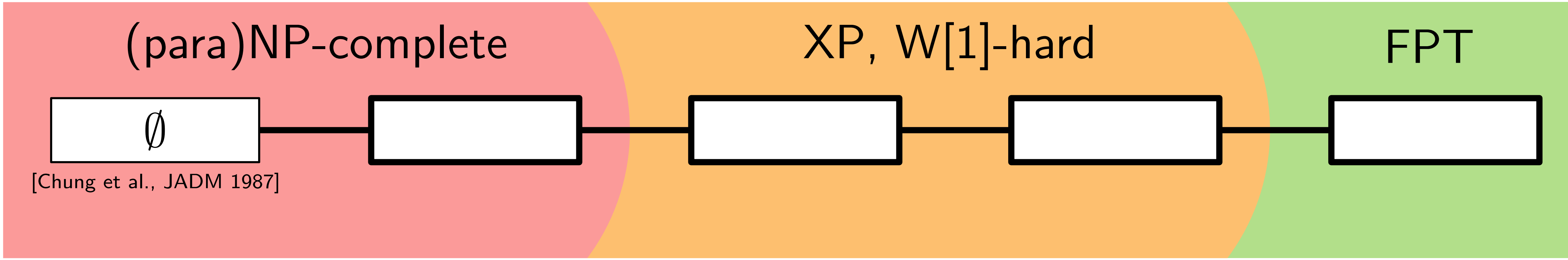
Our Results

NP-complete

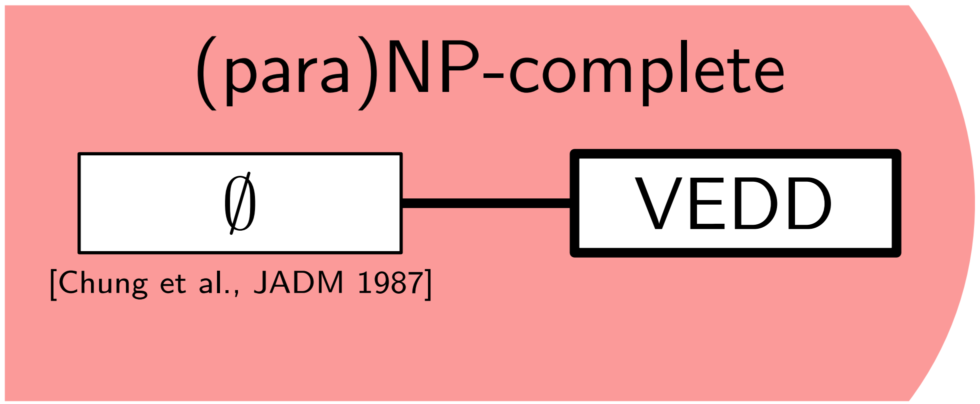
$\emptyset$

[Chung et al., JADM 1987]

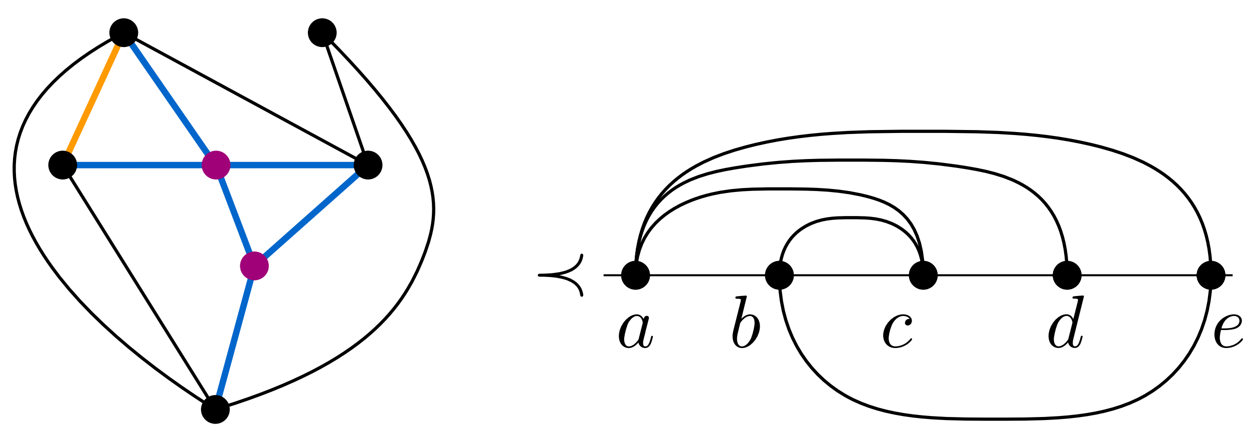
Our Results



Our Results



VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H



Our Results

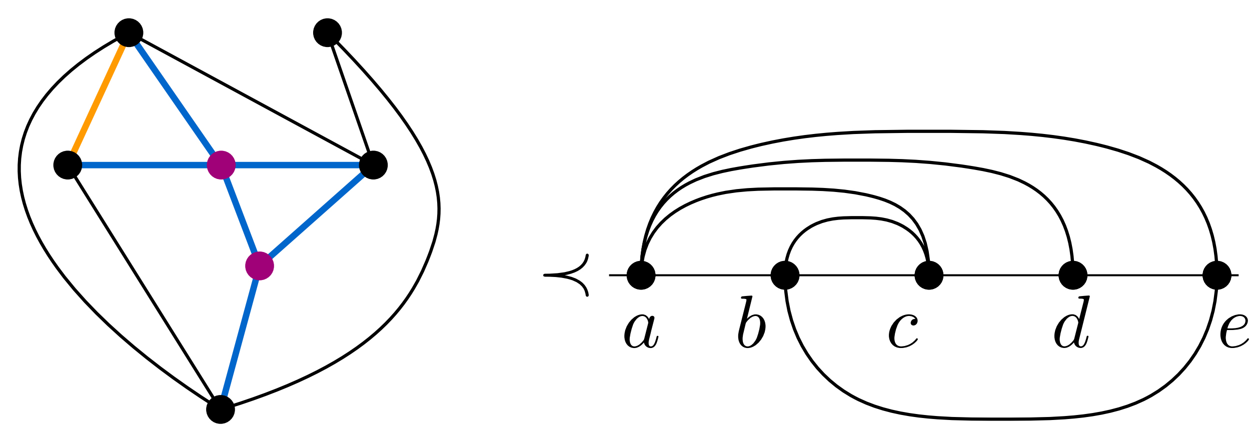
(para)NP-complete

∅ — **VEDD**

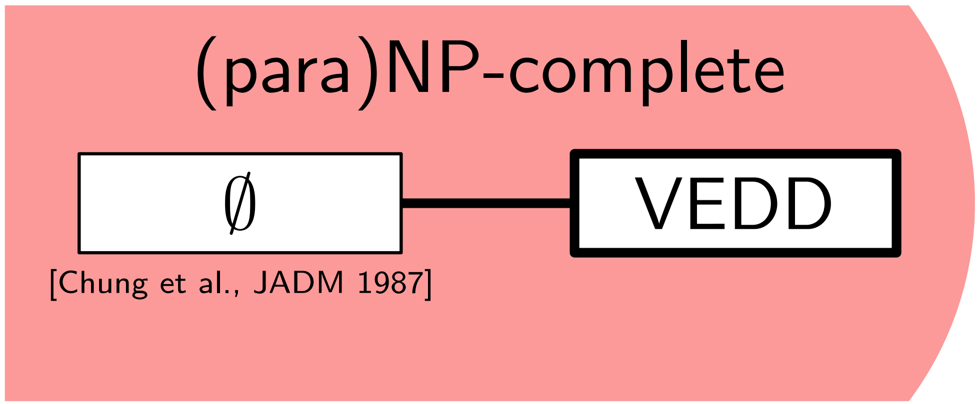
[Chung et al., JADM 1987]

VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H

$\text{VEDD} = \mathbf{2} + \mathbf{1} = 3$



Our Results

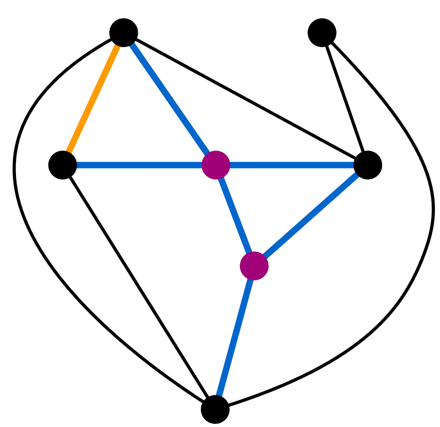


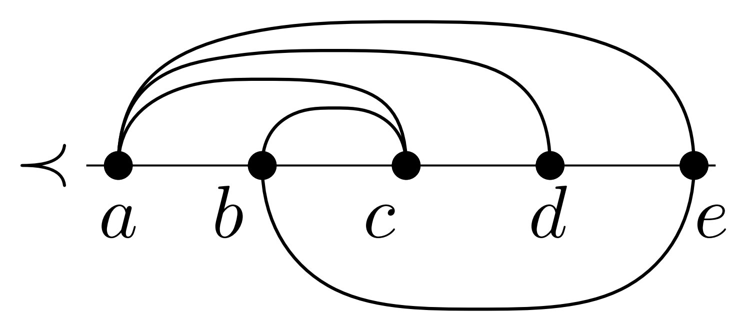
VEDD was successfully used in other drawing extension problems

[Bhore et al., SoCG 2020] [Eiben et al., MFCS 2020]
[Eiben et al., ICALP 2020]

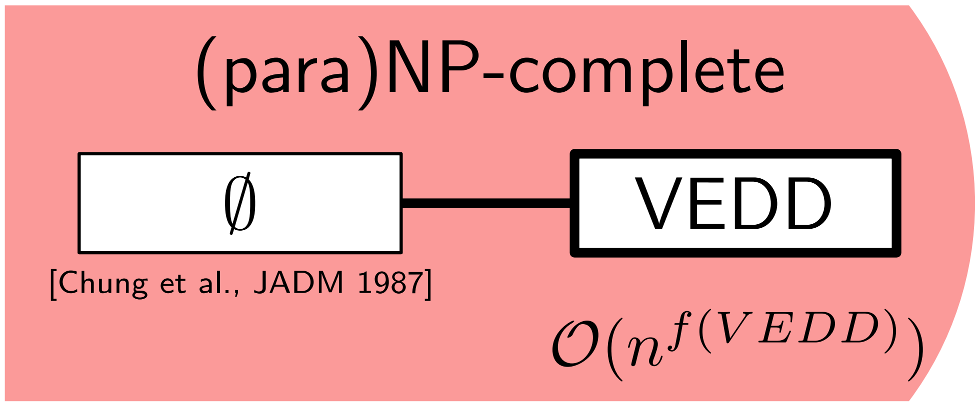
VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H

VEDD $= \mathbf{2} + \mathbf{1} = 3$



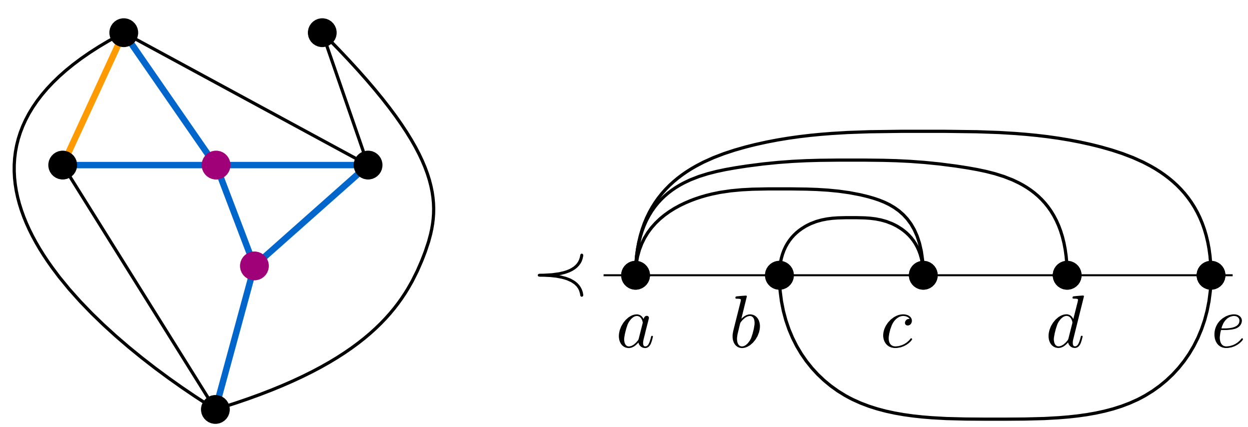


Our Results



VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H

VEDD $= 2 + 1 = 3$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-Sat

$$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \dots$$

SLE With Two Missing Vertices is NP-complete

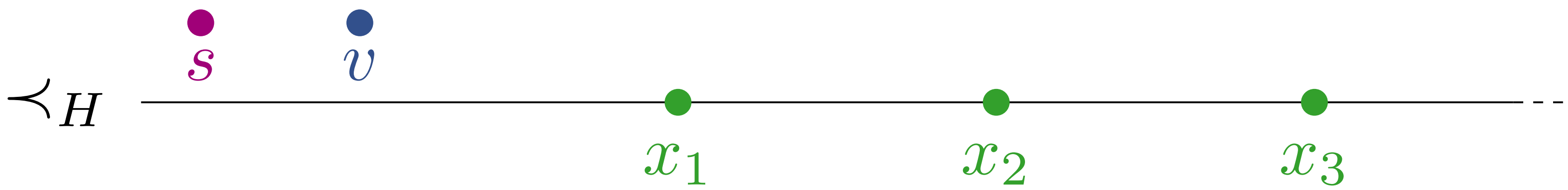
Reduction from 3-SAT

$$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \ldots$$

SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

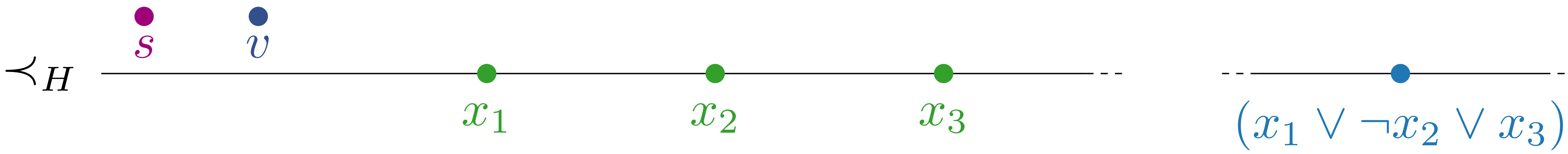
$$\varphi = (x_1 \lor \neg x_2 \lor x_3) \land \ldots$$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

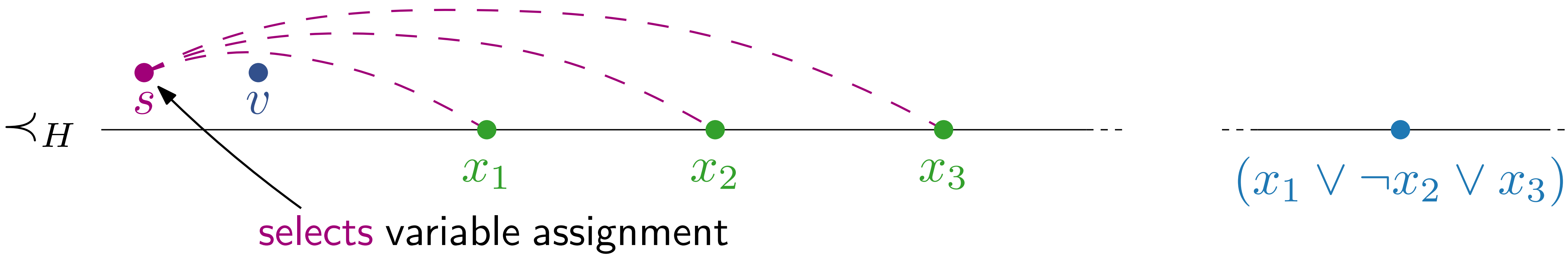
$$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \ldots$$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

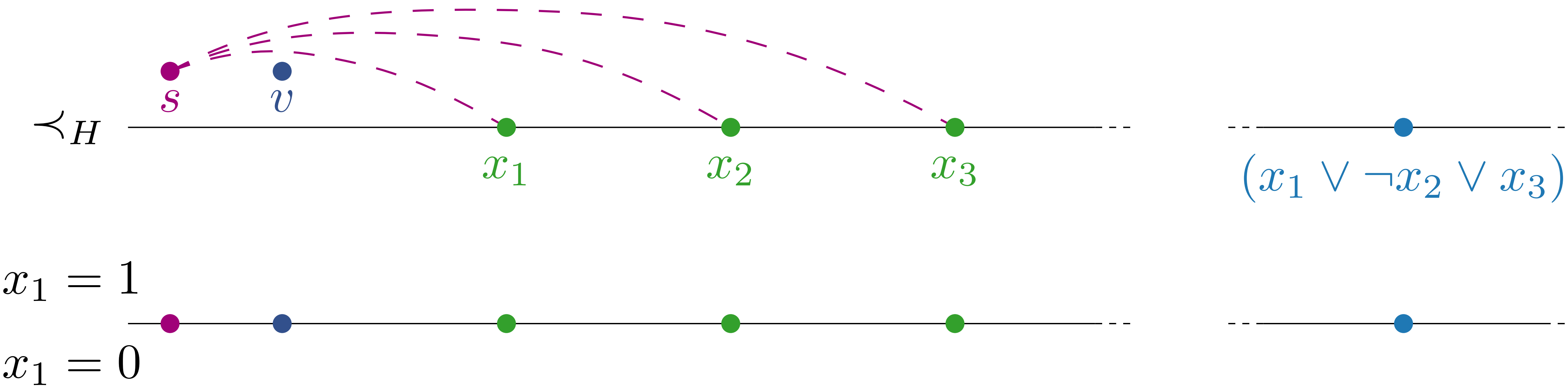
$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \dots$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

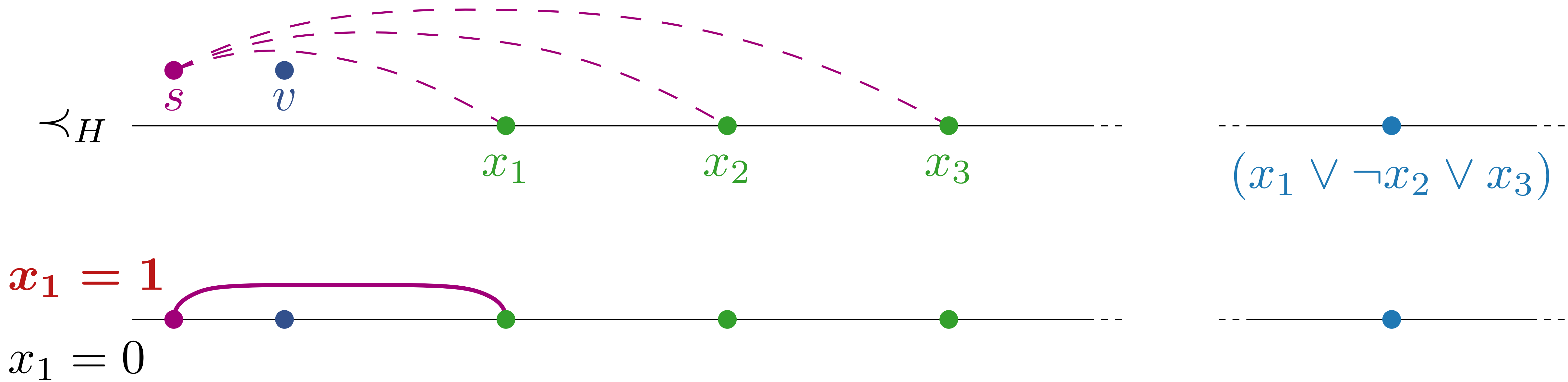
$\varphi = (x_1 \lor \neg x_2 \lor x_3) \land \ldots$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

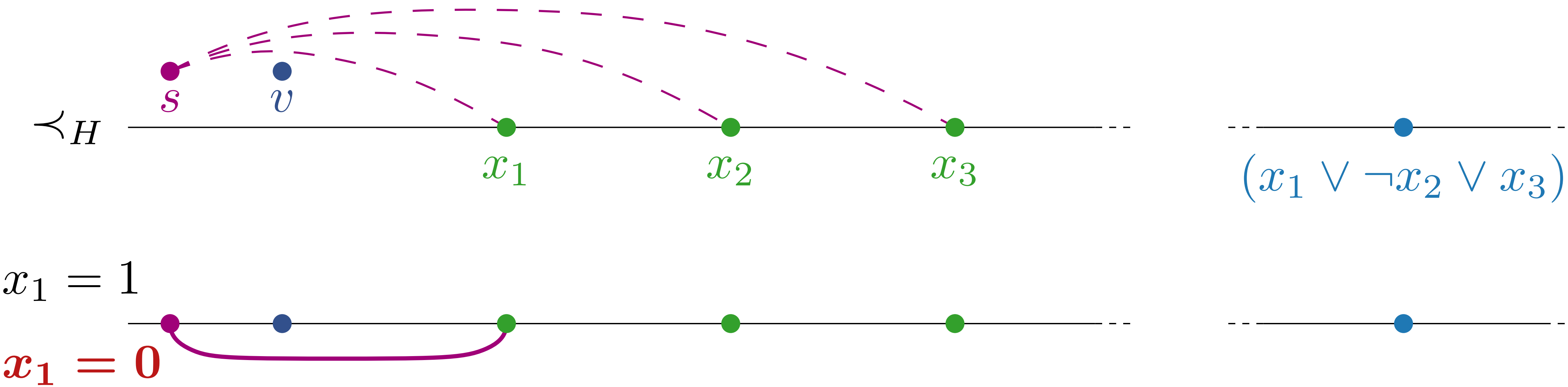
$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \dots$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

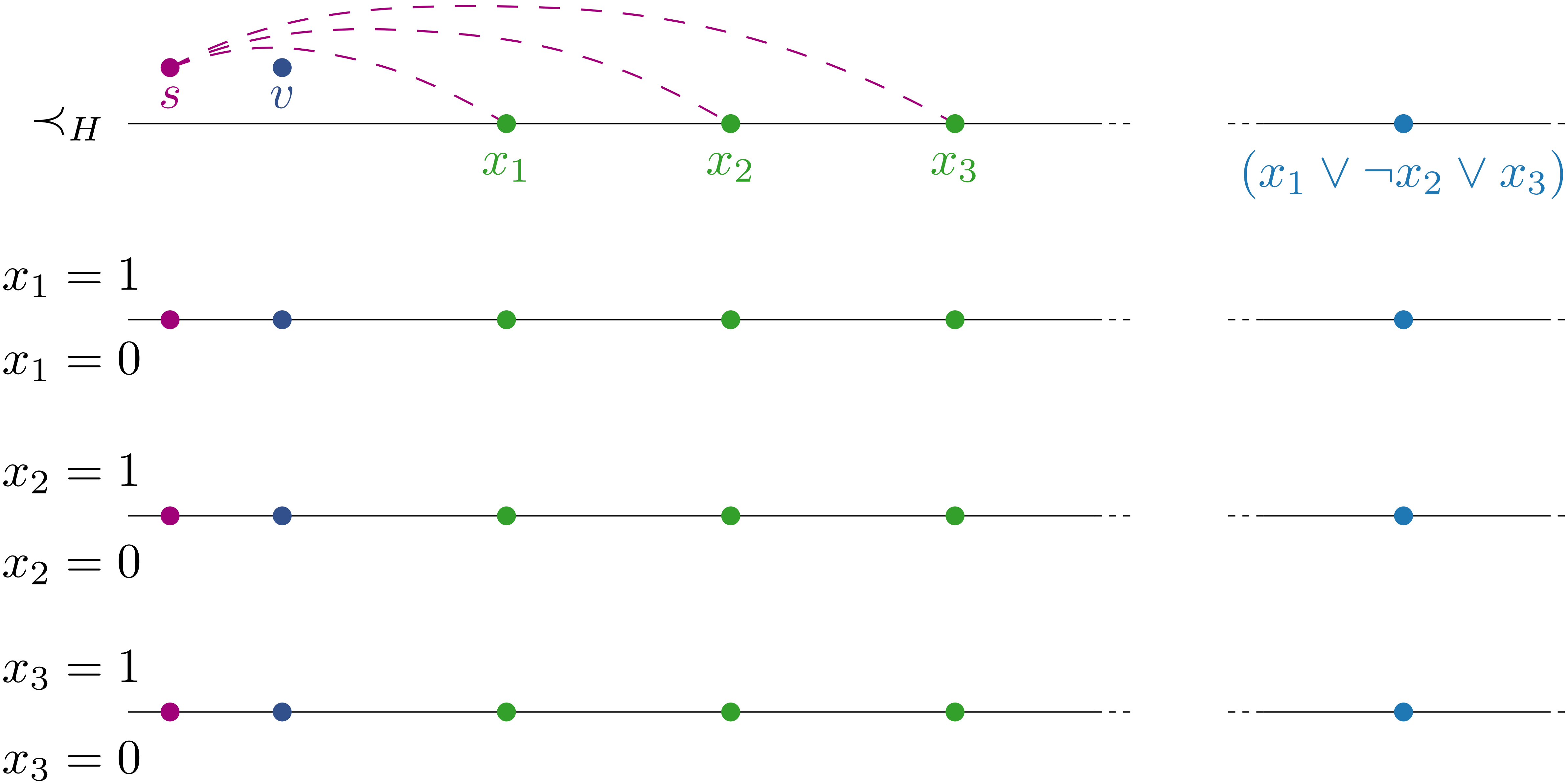
$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \ldots$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

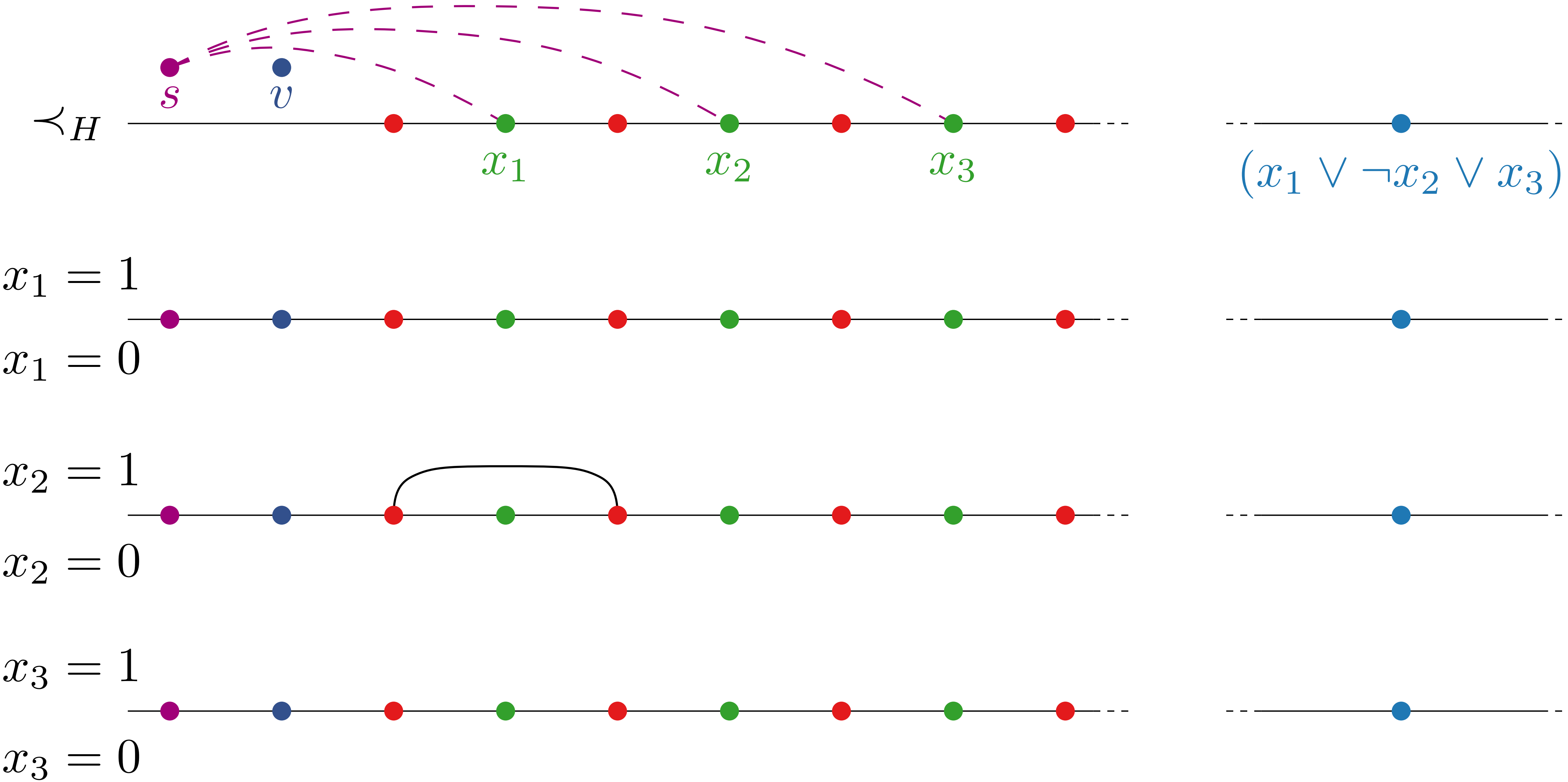
$$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \dots$$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-Sat

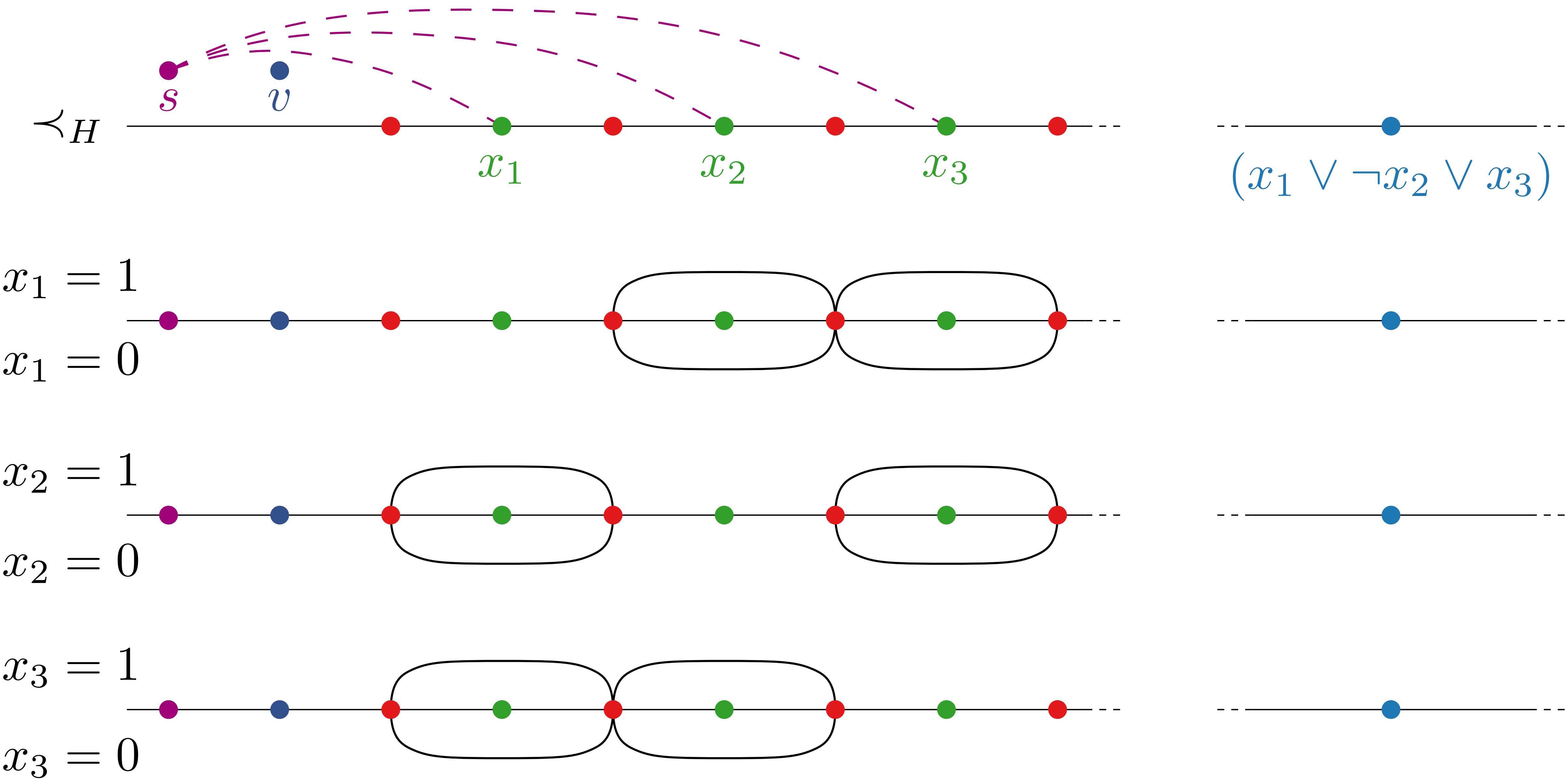
$$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \ldots$$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-Sat

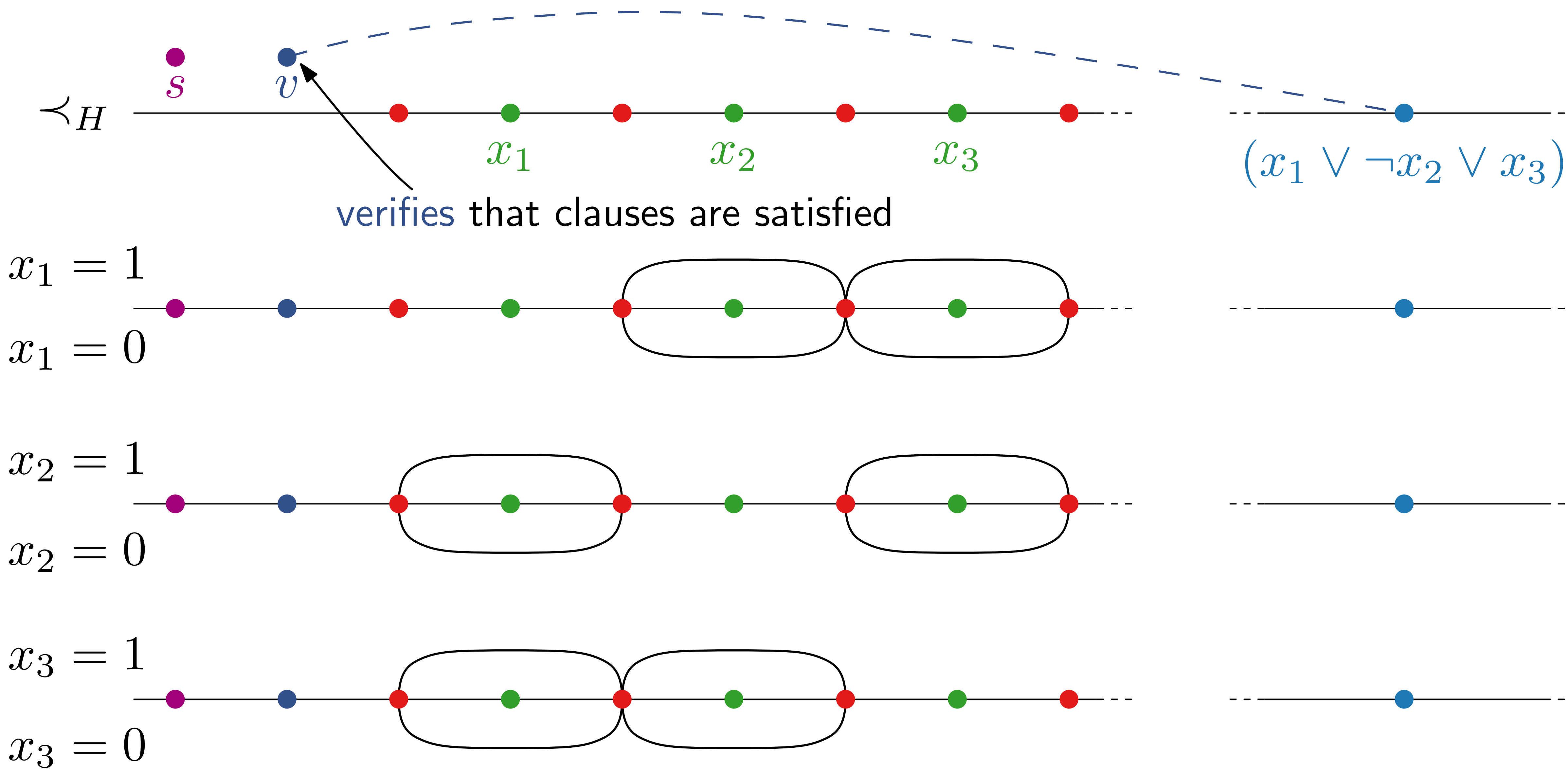
$\varphi = (x_1 \lor \neg x_2 \lor x_3) \land \ldots$



SLE With Two Missing Vertices is NP-complete

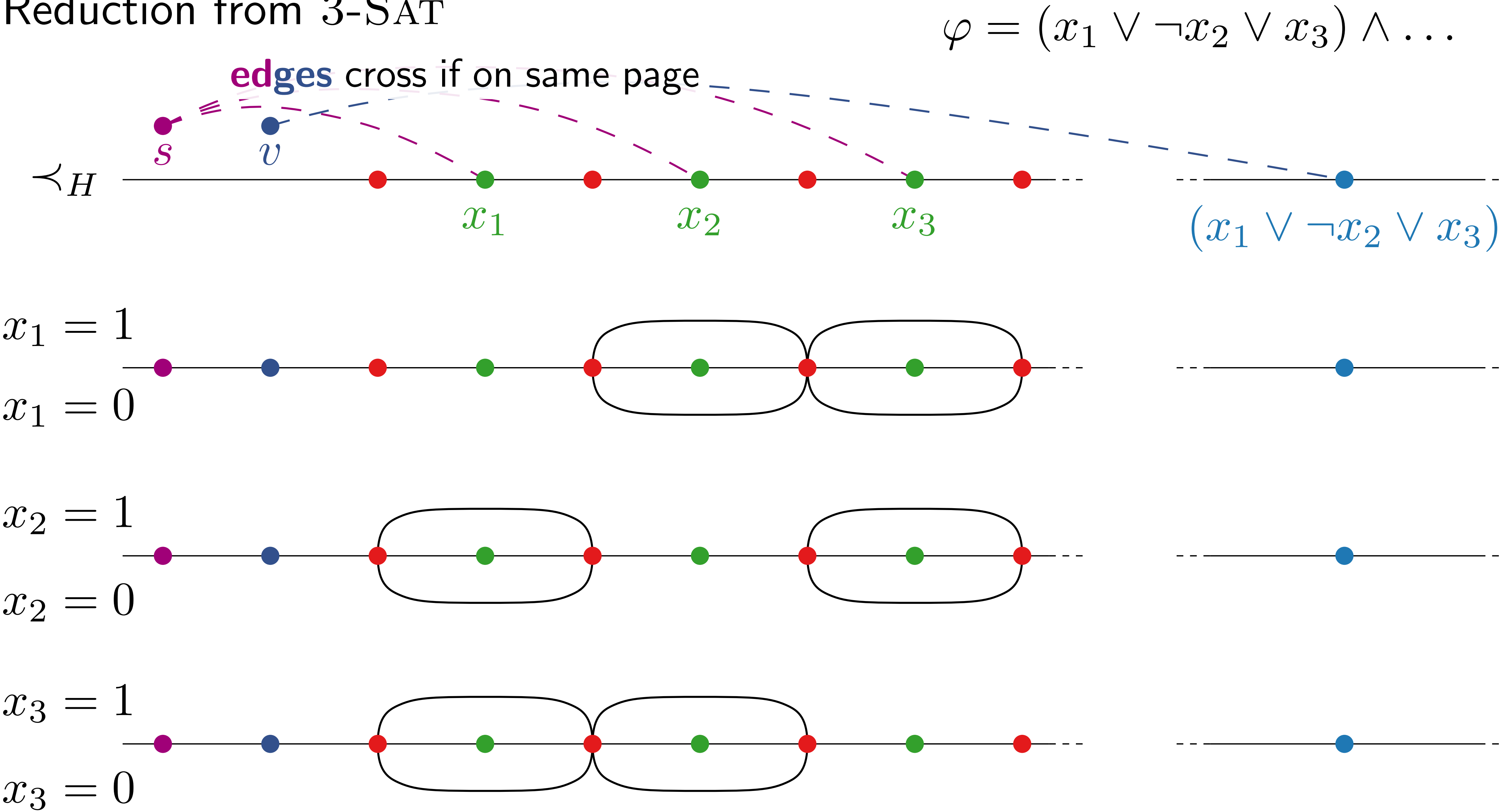
Reduction from 3-SAT

$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \dots$



SLE With Two Missing Vertices is NP-complete

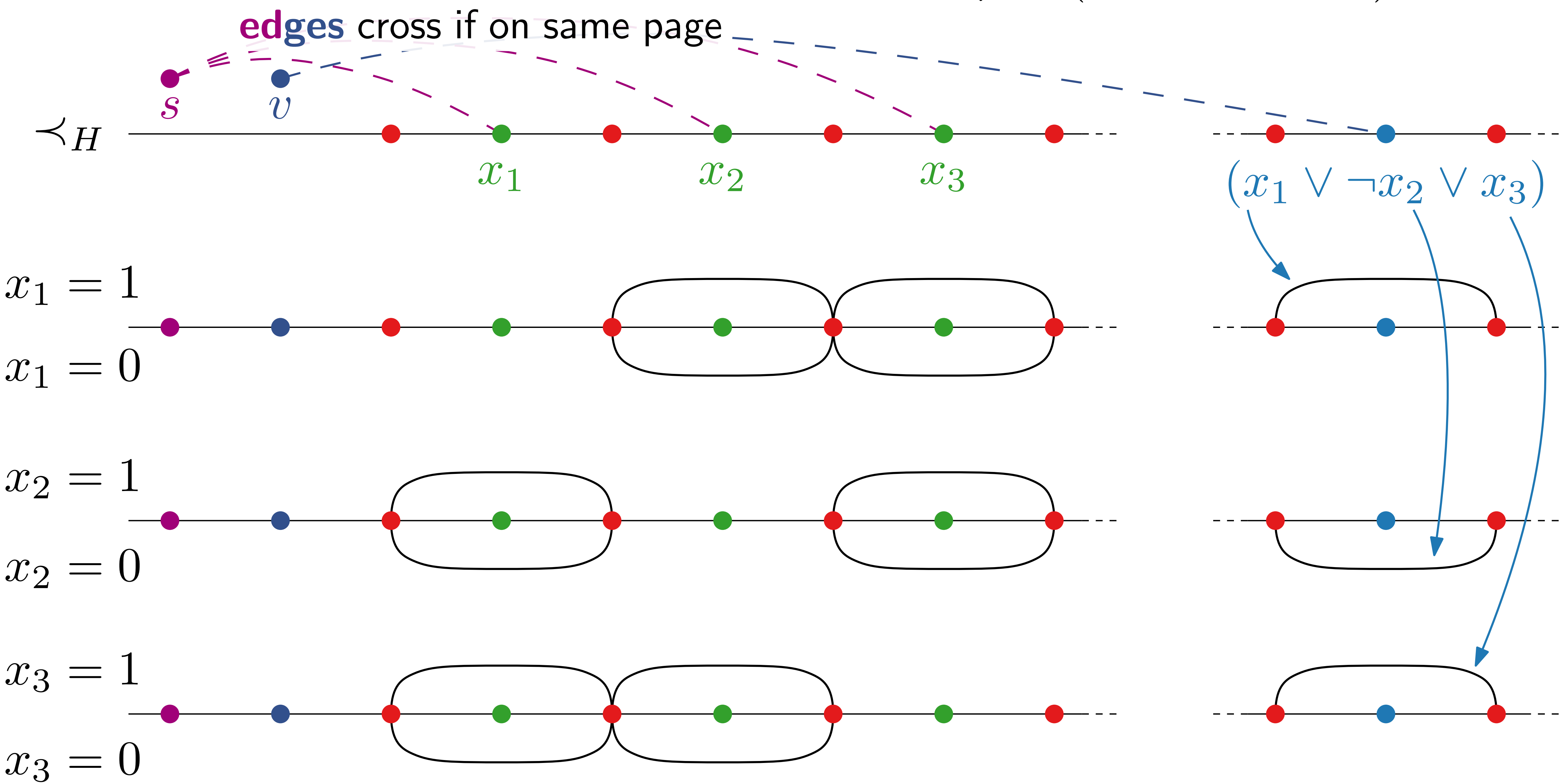
Reduction from 3-SAT



SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

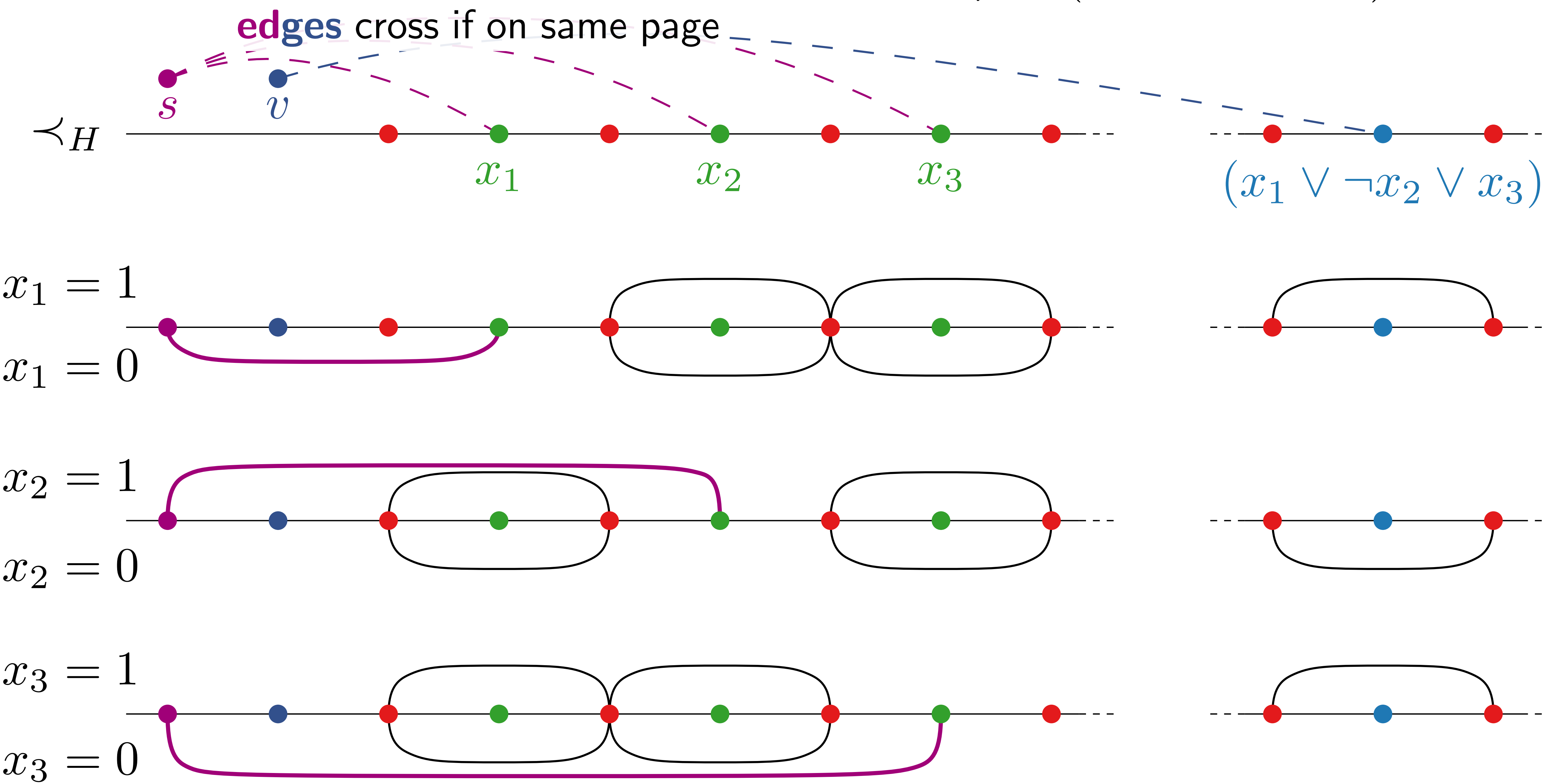
$$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \dots$$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-Sat

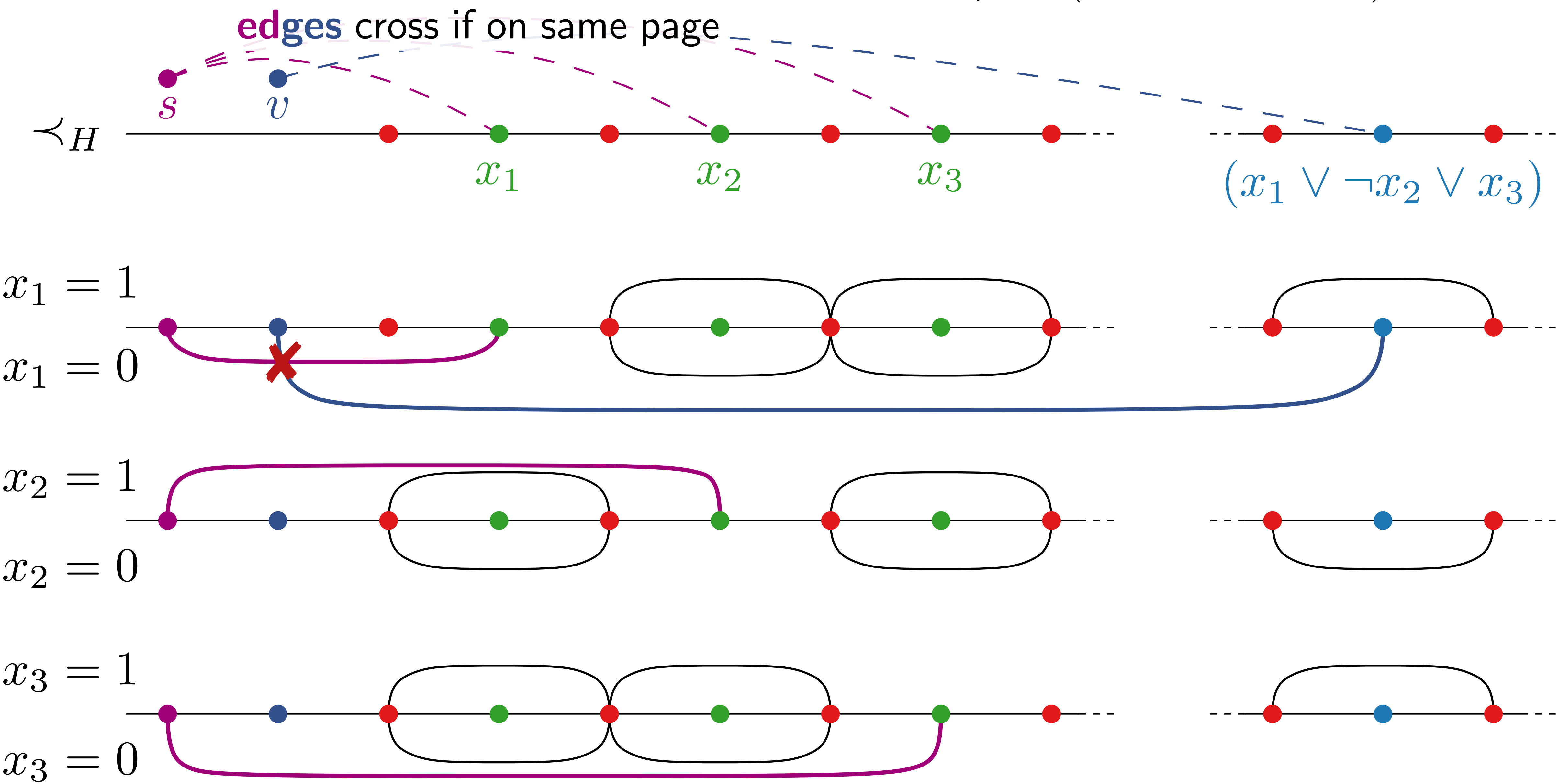
$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \ldots$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

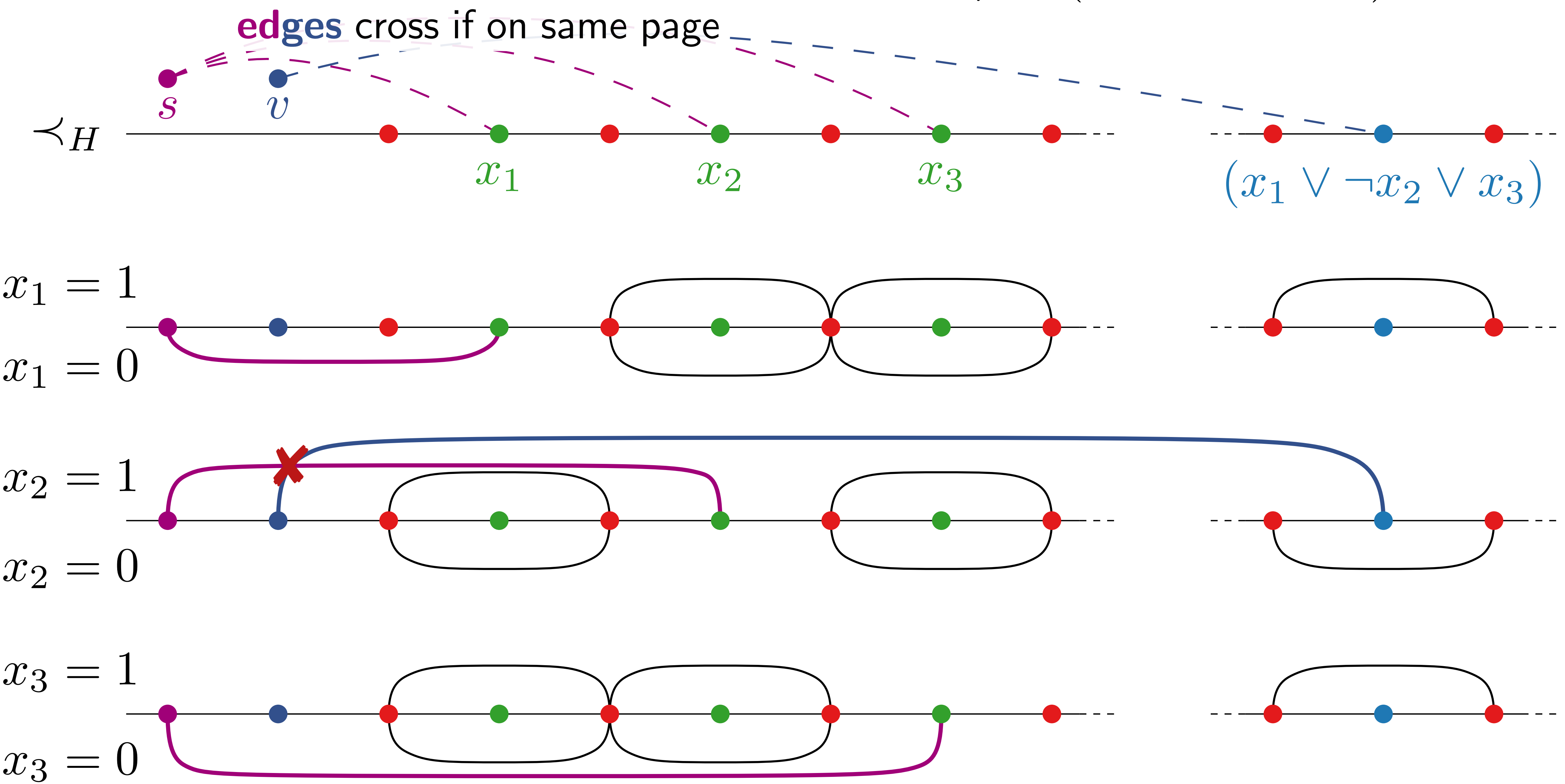
$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \ldots$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

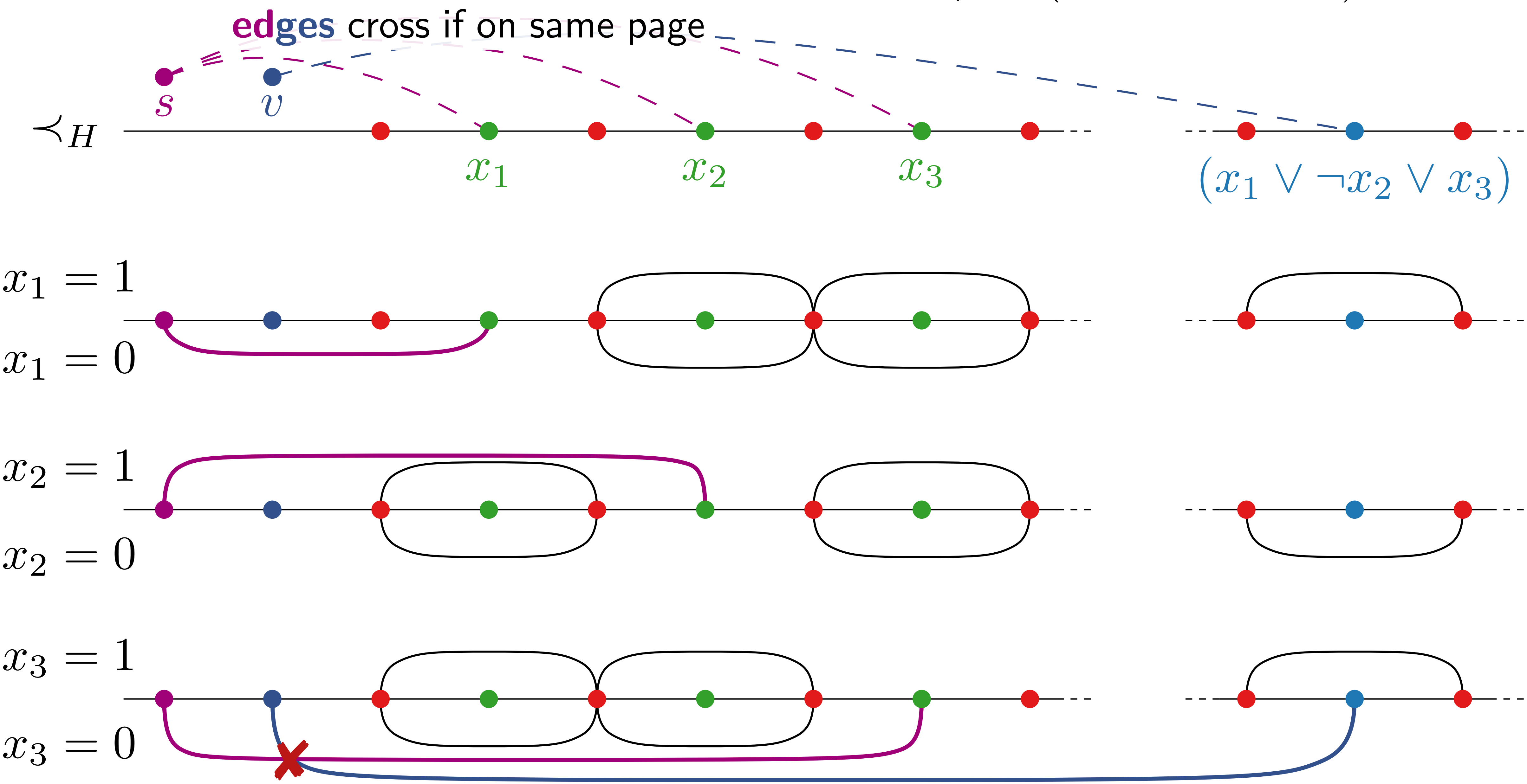
$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \ldots$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-Sat

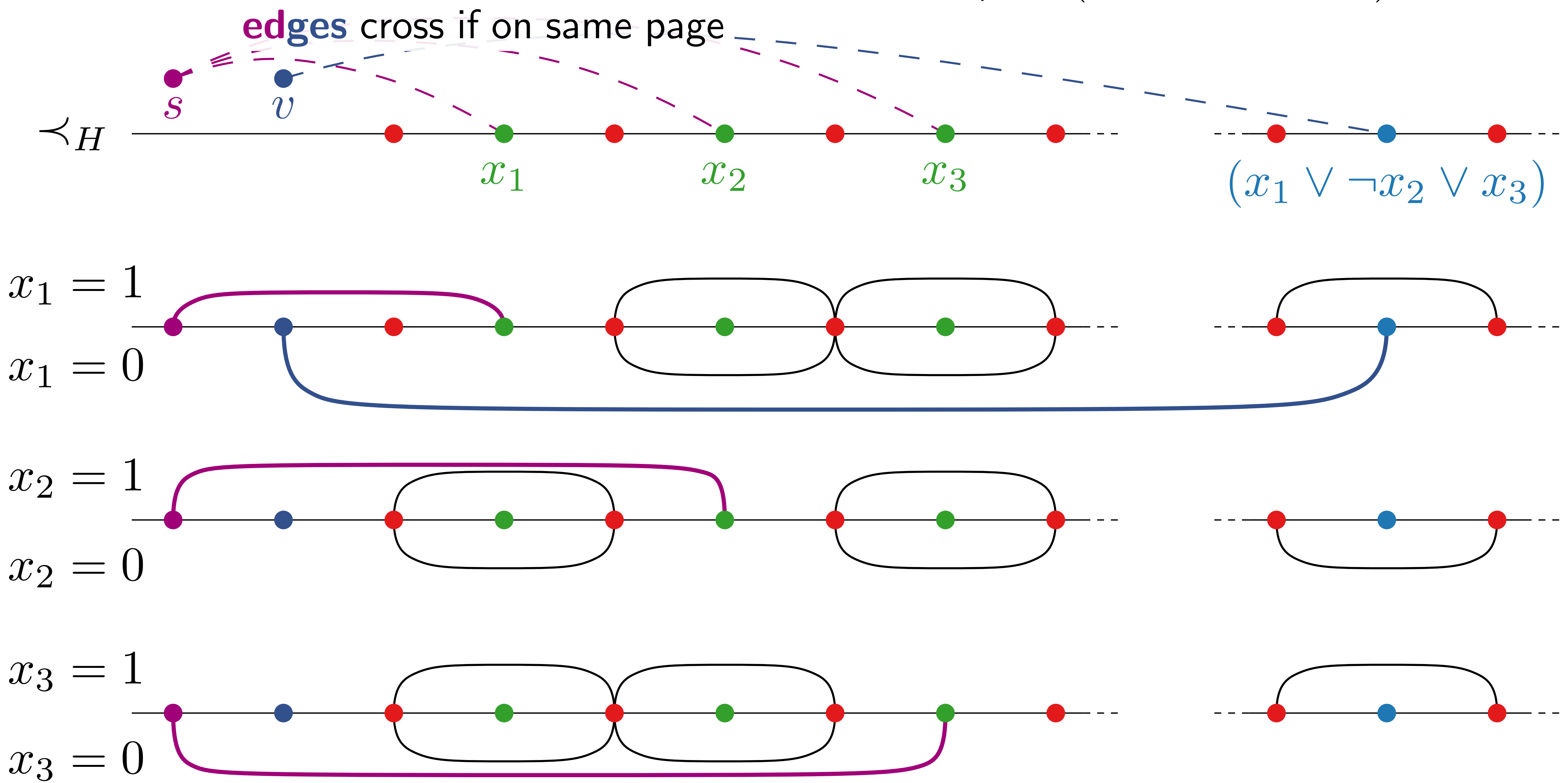
$\varphi = (x_1 \lor \neg x_2 \lor x_3) \land \ldots$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

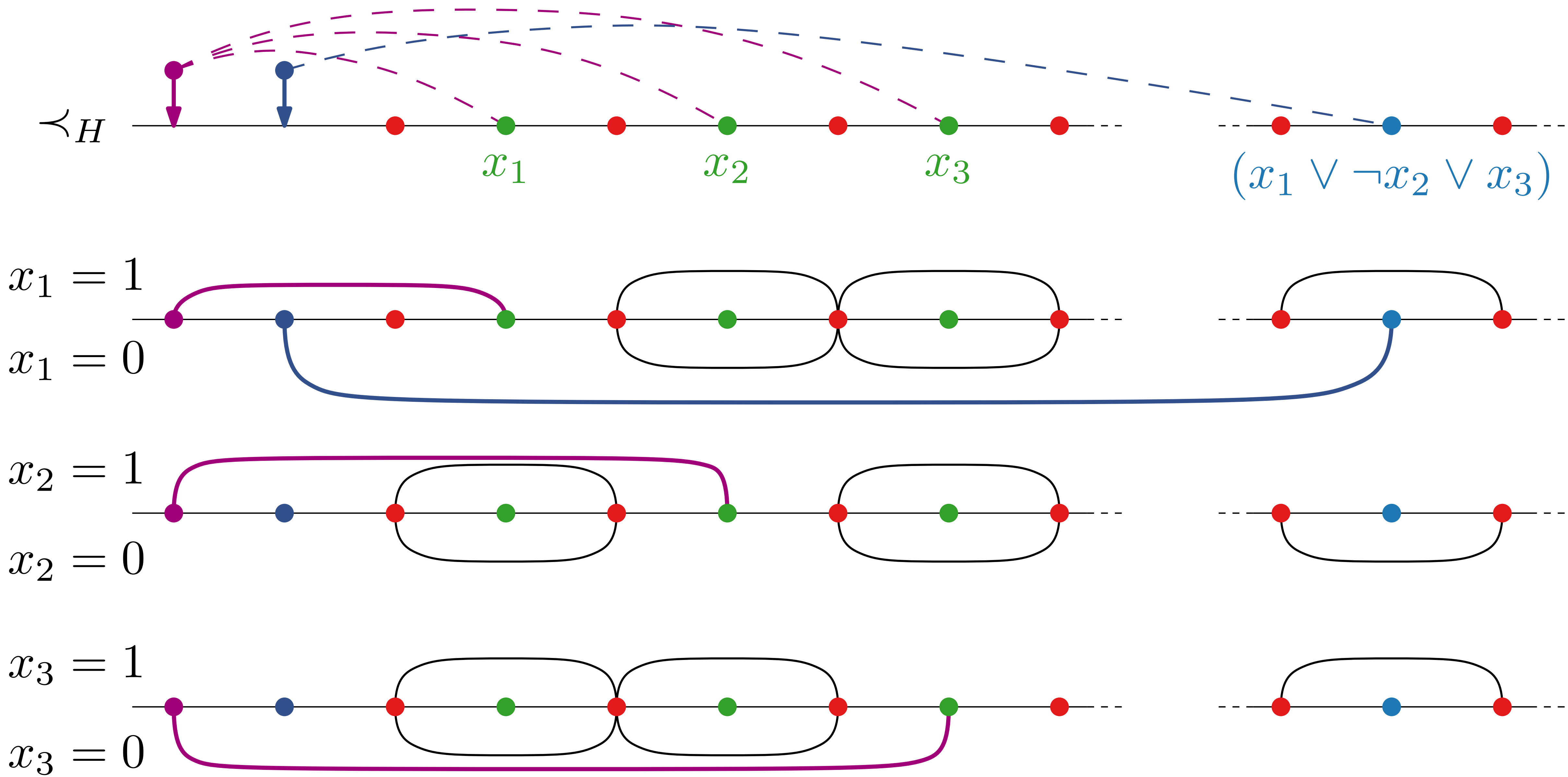
$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \ldots$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

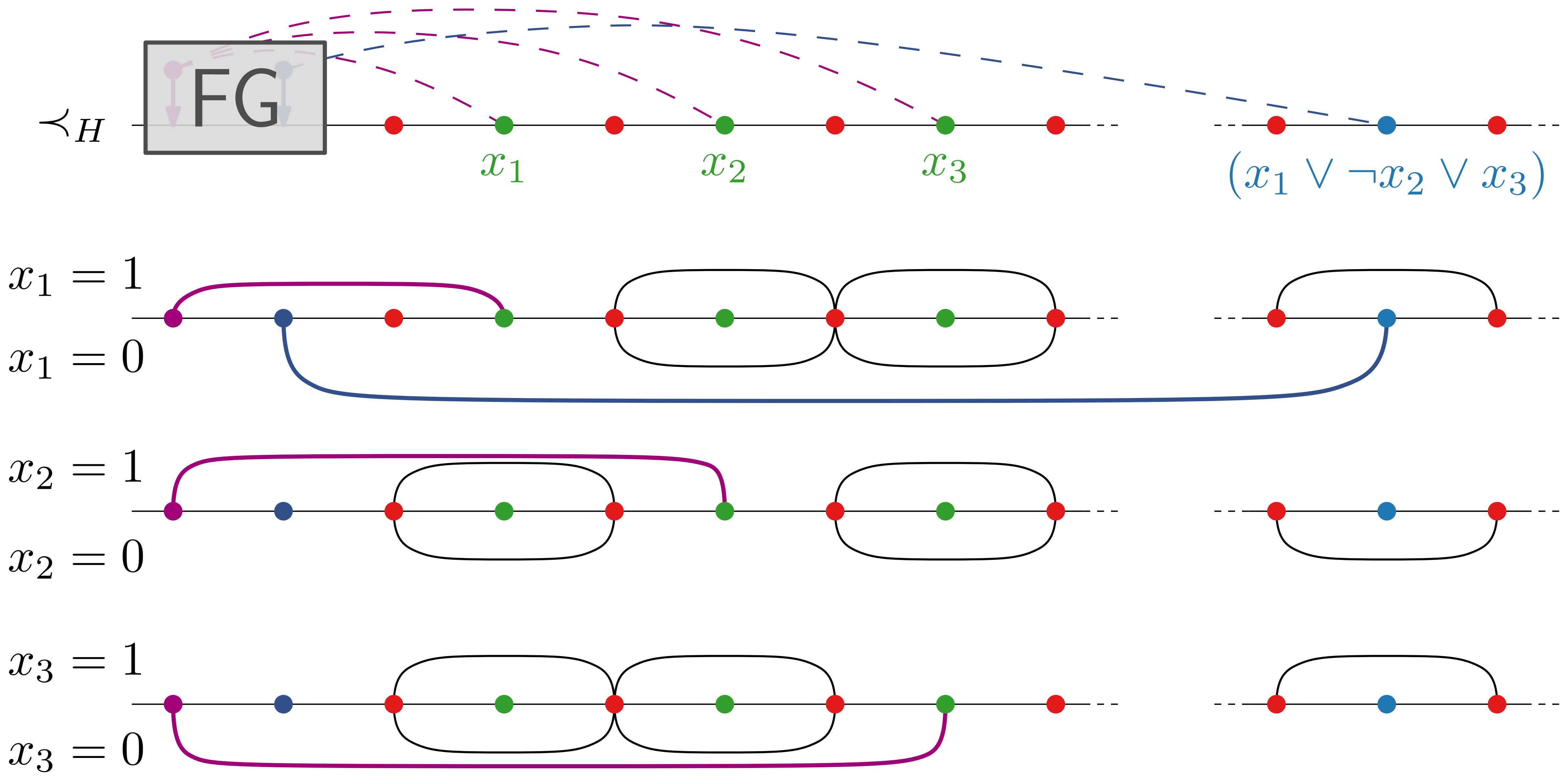
$$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \ldots$$



SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT

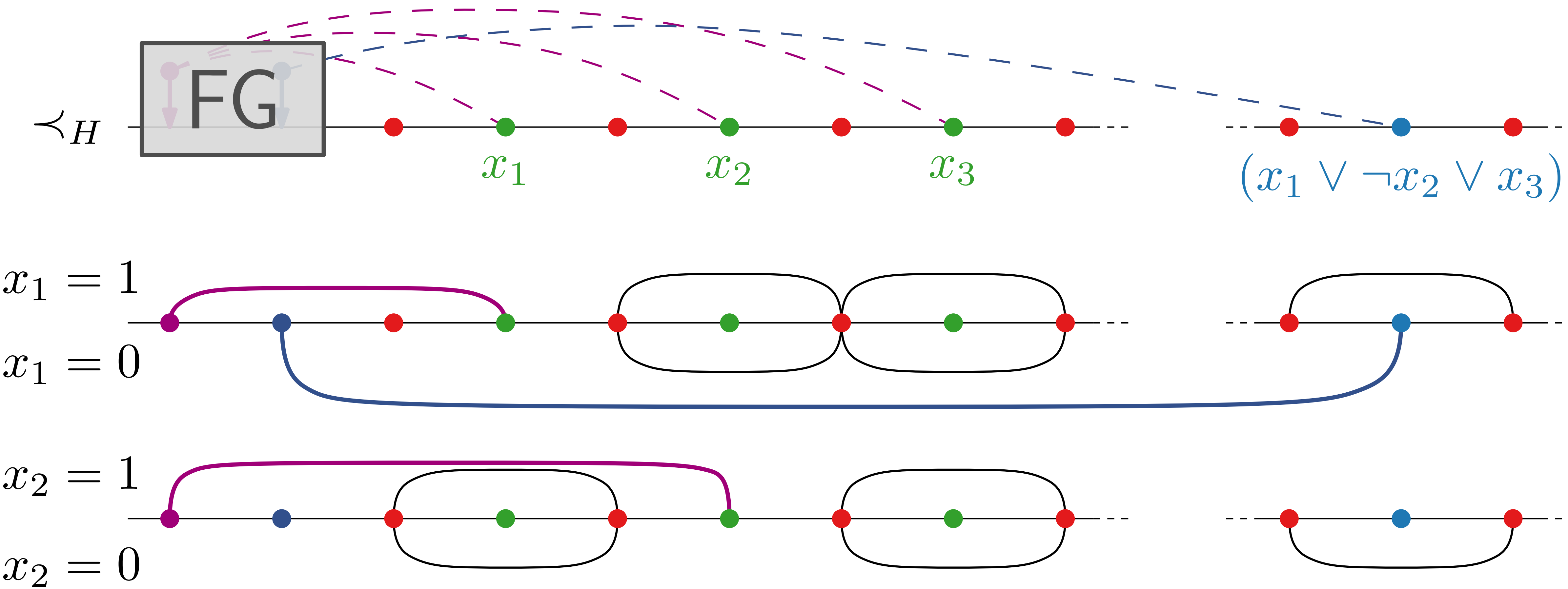
$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \ldots$



SLE With Two Missing Vertices is NP-complete

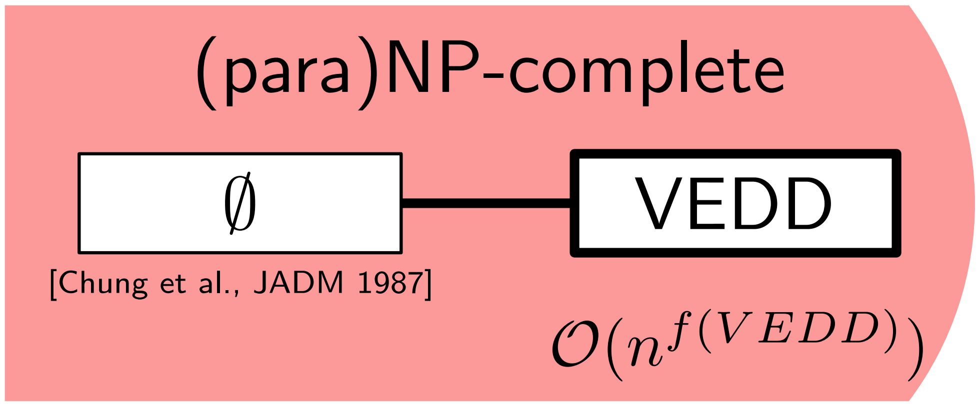
Reduction from 3-SAT

$$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge \dots$$



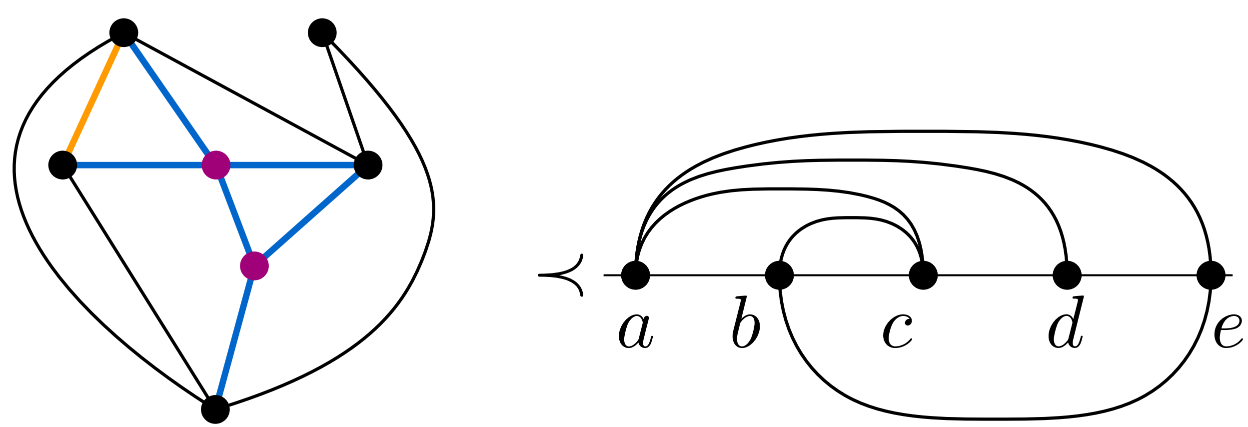
Theorem:
SLE is NP-complete, even for just **two** missing vertices to which all missing edges are incident to.

Our Results



VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H

VEDD $= 2 + 1 = 3$



Our Results

(para)NP-complete

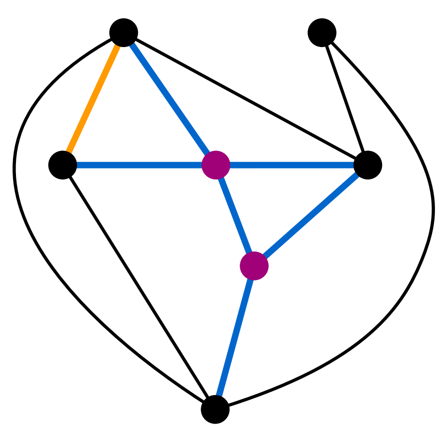
∅ — VEDD

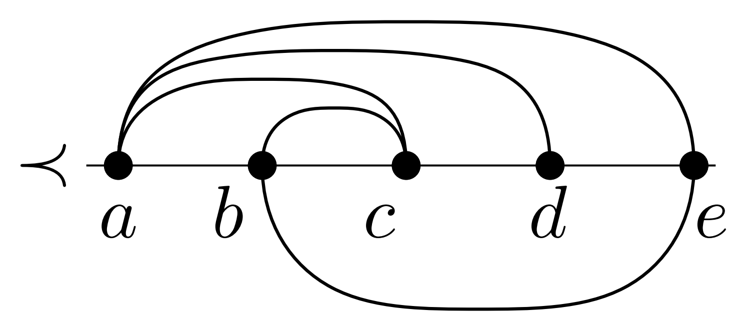
[Chung et al., JADM 1987]

~~$\mathcal{O}(n^{f(VEDD)})$~~

VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H

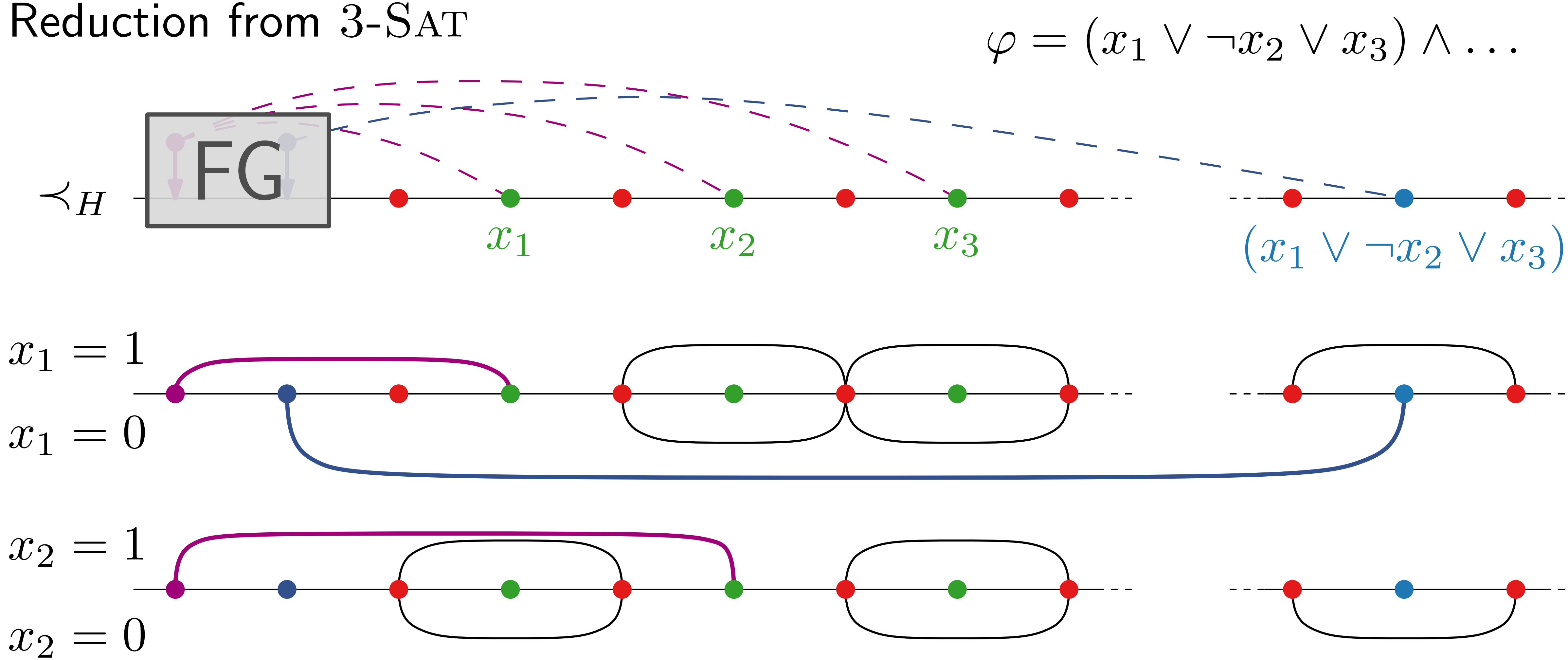
VEDD = **2** + **1** = 3





Rewind: SLE With Two Missing Vertices is NP-complete

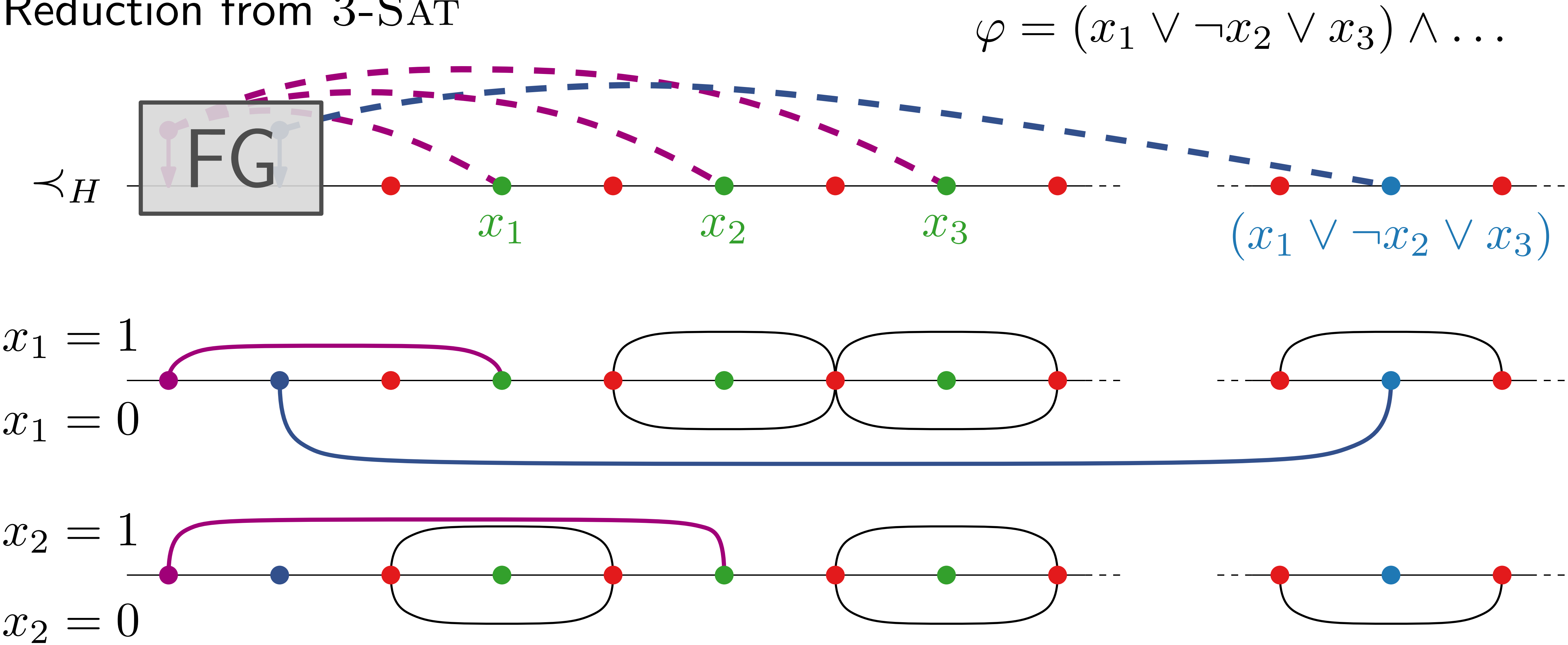
Reduction from 3-SAT



Theorem:
SLE is NP-complete, even for just **two** missing vertices to which all missing edges are incident to.

Rewind: SLE With Two Missing Vertices is NP-complete

Reduction from 3-SAT



Theorem:
SLE is NP-complete, even for just **two** missing vertices to which all missing edges are incident to.

Our Results

(para)NP-complete

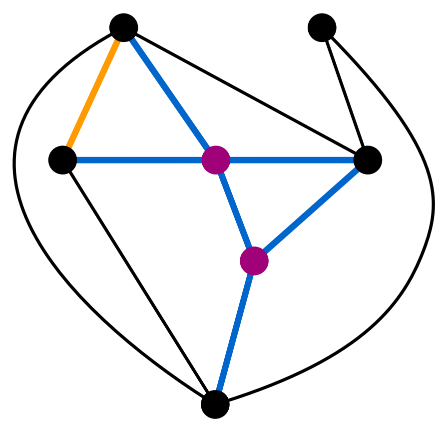
∅ — VEDD

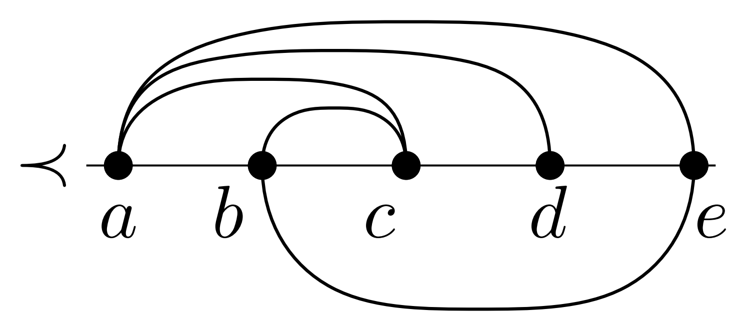
[Chung et al., JADM 1987]

~~$\mathcal{O}(n^{f(VEDD)})$~~

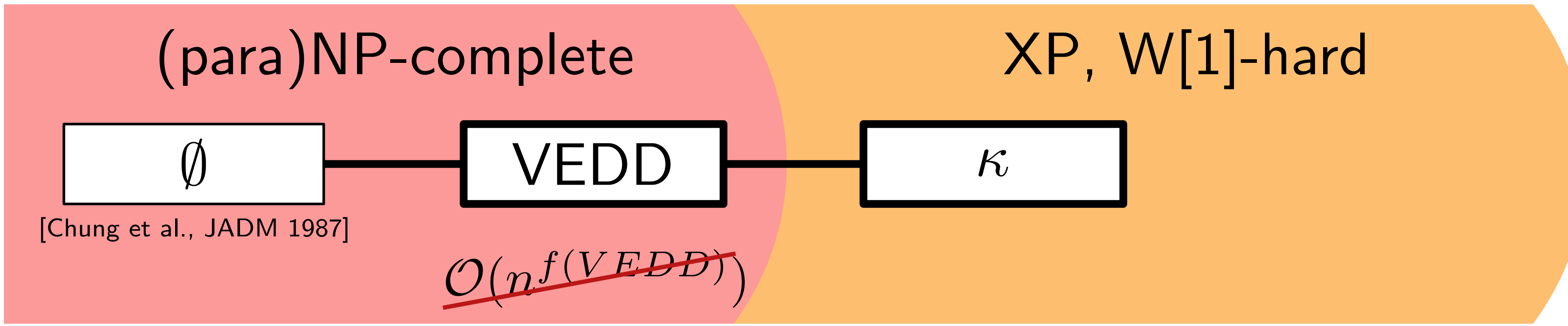
VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H

VEDD $= 2 + 1 = 3$



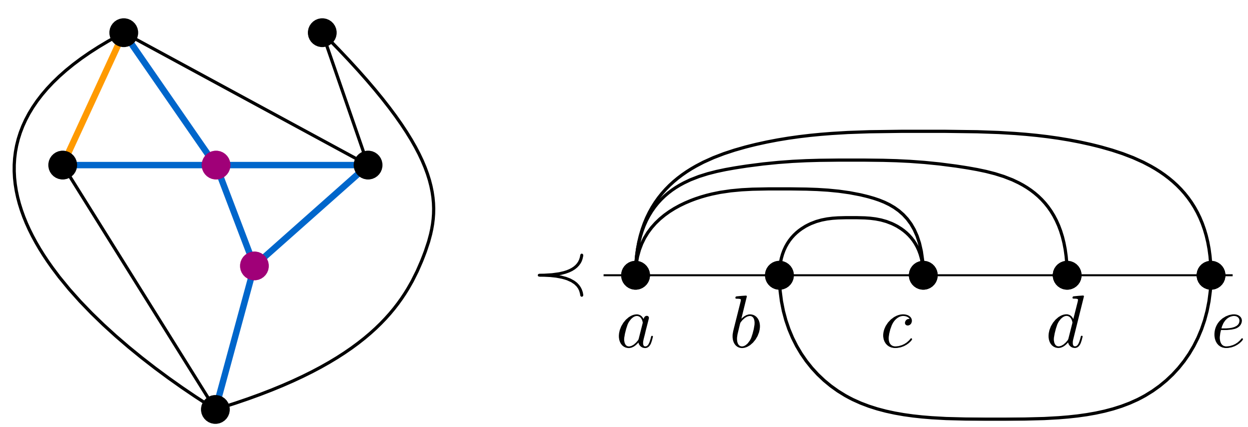


Our Results

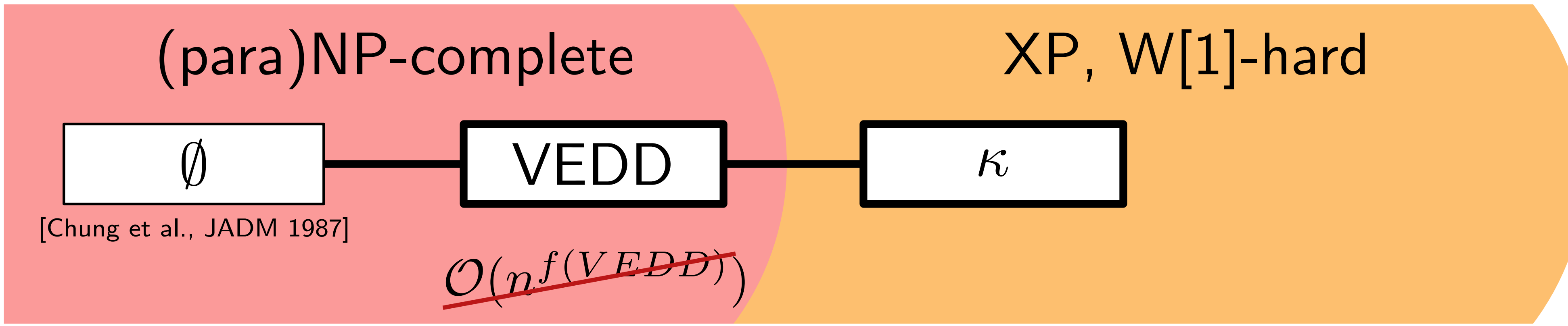


VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H
κ: #missing vertices **plus** #missing edges

$\text{VEDD} = 2 + 1 = 3$



Our Results

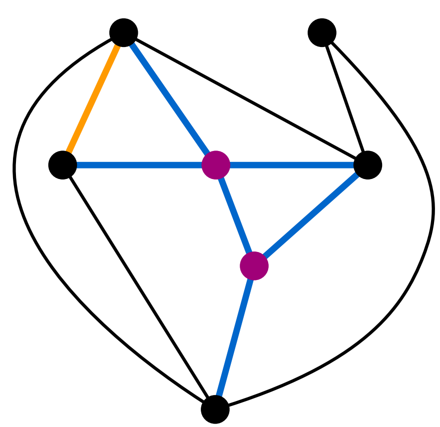


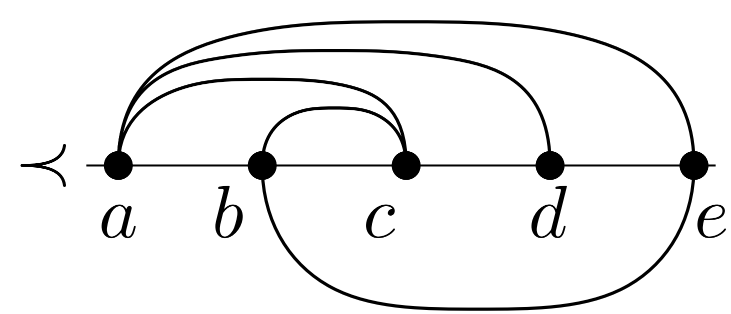
VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H

κ: #missing vertices **plus** #missing edges

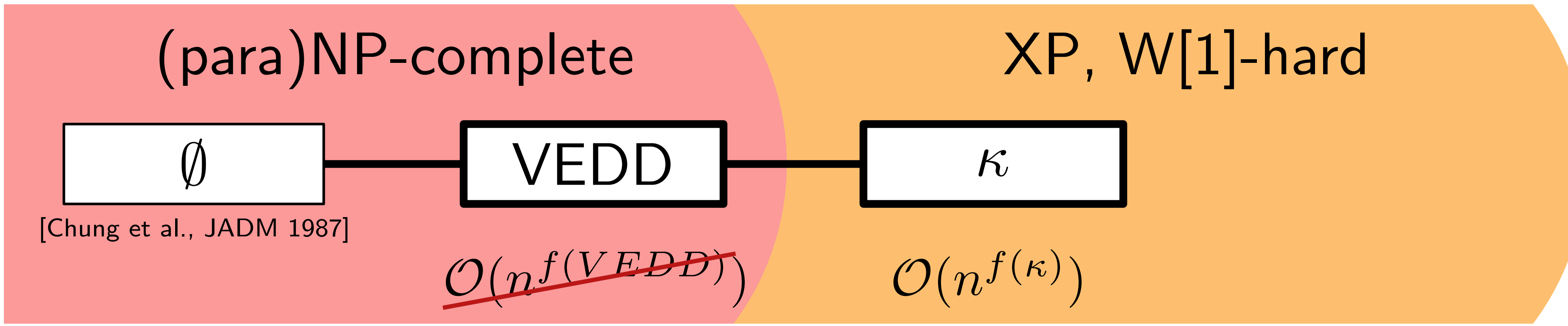
$\text{VEDD} = \mathbf{2} + \mathbf{1} = 3$

$\kappa = \mathbf{2} + \mathbf{1} + \mathbf{6} = 9$





Our Results

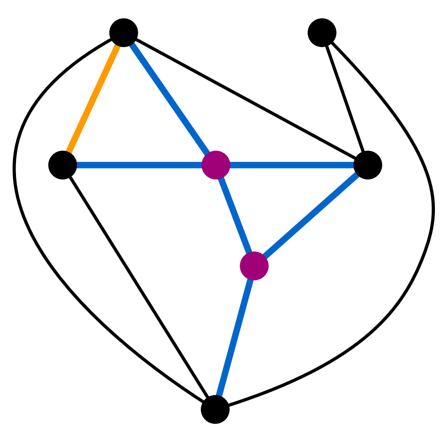


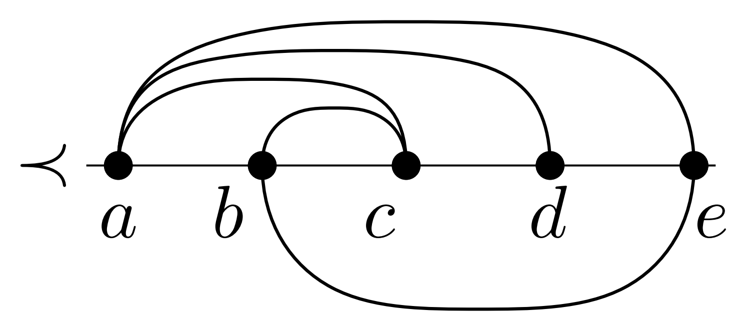
VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H

κ: #missing vertices **plus** #missing edges

$\text{VEDD} = \mathbf{2} + \mathbf{1} = 3$

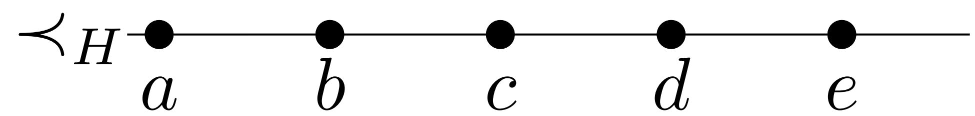
$\kappa = \mathbf{2} + \mathbf{1} + \mathbf{6} = 9$





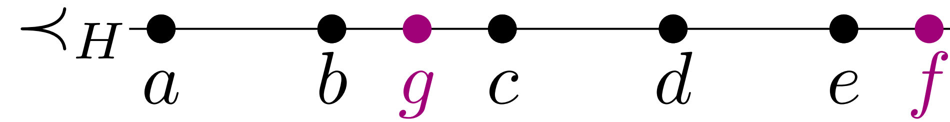
SLE Parameterized by κ is in XP

Step 1: Guess the extended spine order $\prec_G$



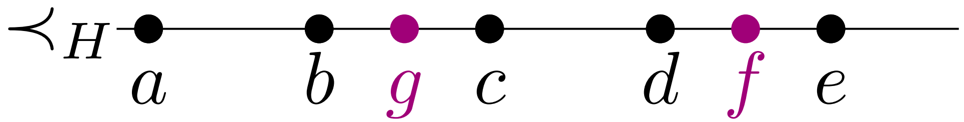
SLE Parameterized by κ is in XP

Step 1: Guess the extended spine order $\prec_G$



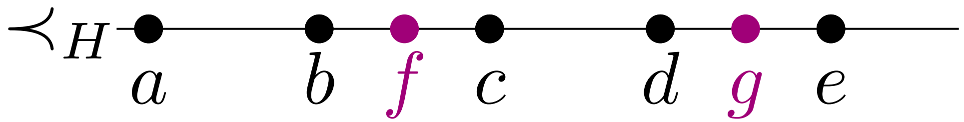
SLE Parameterized by κ is in XP

Step 1: Guess the extended spine order $\prec_G$



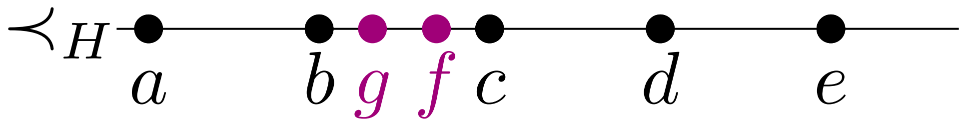
SLE Parameterized by κ is in XP

Step 1: Guess the extended spine order $\prec_G$



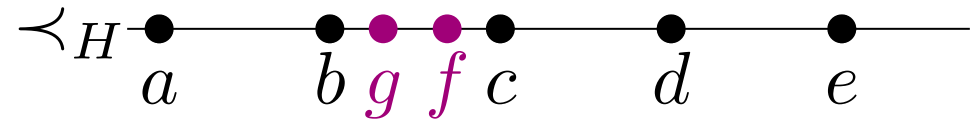
SLE Parameterized by κ is in XP

Step 1: Guess the extended spine order $\prec_G$



SLE Parameterized by κ is in XP

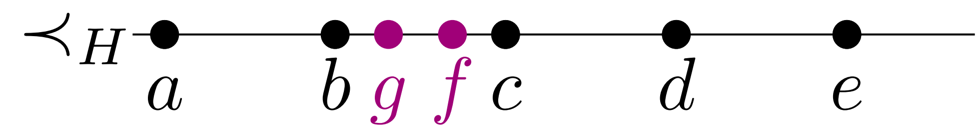
Step 1: Guess the extended spine order $\prec_G$



Step 2: Solve special instance of SLE where only edges are missing

SLE Parameterized by κ is in XP

Step 1: Guess the extended spine order $\prec_G$

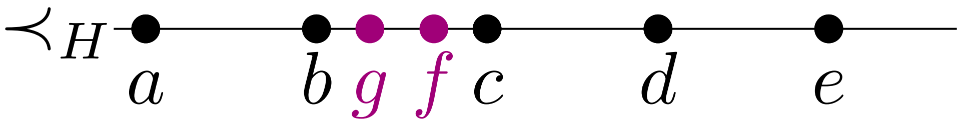


Step 2: Solve special instance of SLE where only edges are missing

$\mathcal{O}({m_{add}}^{m_{add}} \cdot |\mathcal{I}|)$ time

SLE Parameterized by κ is in XP

Step 1: Guess the extended spine order $\prec_G$



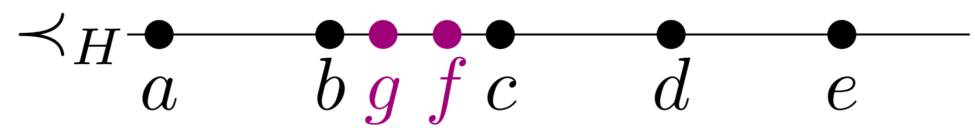
Step 2: Solve special instance of SLE where only edges are missing

$\mathcal{O}({m_{add}}^{m_{add}} \cdot |\mathcal{I}|)$ time

Theorem:
SLE can be solved in $\mathcal{O}(|\mathcal{I}|^{n_{\text{add}}+1} \cdot {m_{\text{add}}}^{m_{\text{add}}})$ time.

SLE Parameterized by κ is in XP

Step 1: Guess the extended spine order $\prec_G$



Step 2: Solve special instance of SLE where only edges are missing

$\mathcal{O}({m_{add}}^{m_{add}} \cdot |\mathcal{I}|)$ time

Size of instance

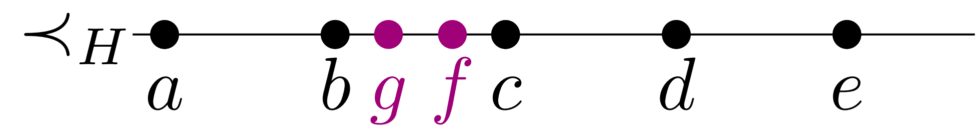
#missing vertices

#missing edges

Theorem:
SLE can be solved in $\mathcal{O}(|\mathcal{I}|^{n_{\text{add}}+1} \cdot {m_{\text{add}}}^{m_{\text{add}}})$ time.

SLE Parameterized by κ is in XP

Step 1: Guess the extended spine order $\prec_G$

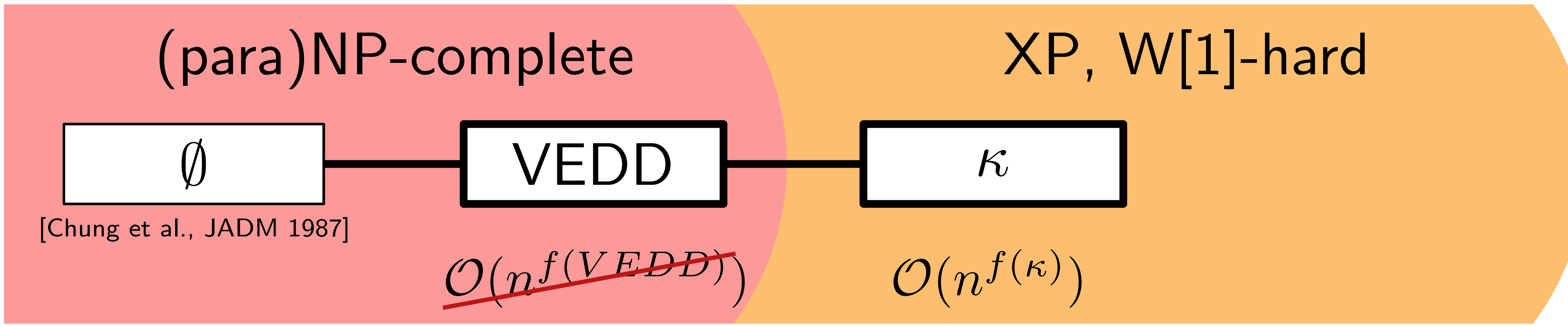


Step 2: Solve special instance of SLE where only edges are missing

$\mathcal{O}({m_{add}}^{m_{add}} \cdot |\mathcal{I}|)$ time

Theorem:
SLE can be solved in $\mathcal{O}(|\mathcal{I}|^{\boldsymbol{f(\kappa)}})$ time.

Our Results

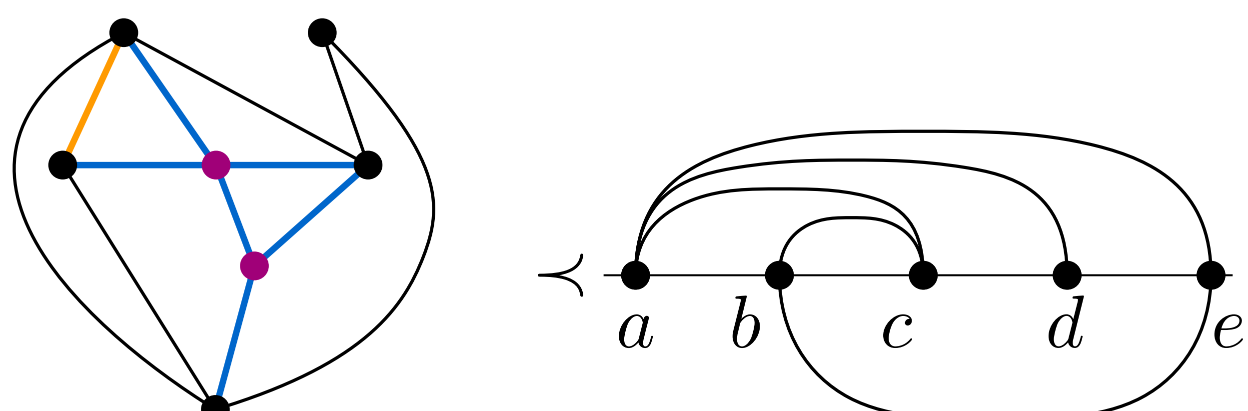


VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H

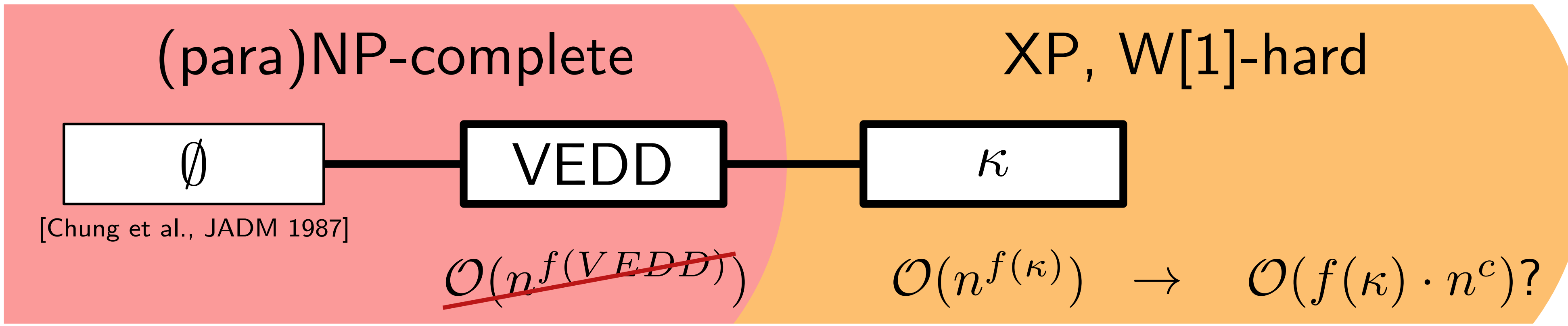
κ: #missing vertices **plus** # missing edges

VEDD $= 2 + 1 = 3$

$\kappa = 2 + 1 + 6 = 9$



Our Results

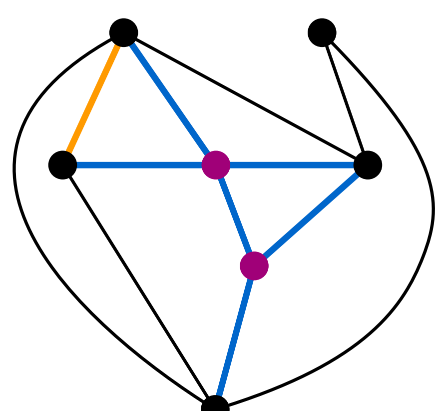


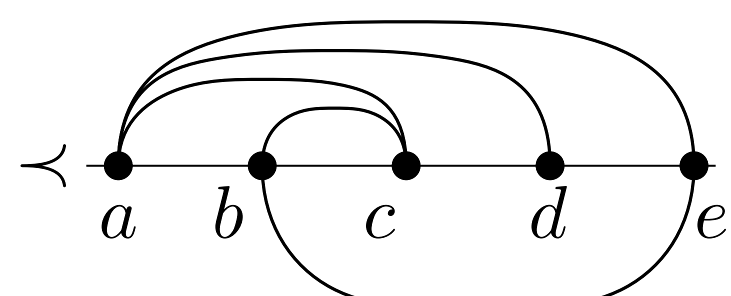
VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H

κ: #missing vertices **plus** # missing edges

$\text{VEDD} = \mathbf{2} + \mathbf{1} = 3$

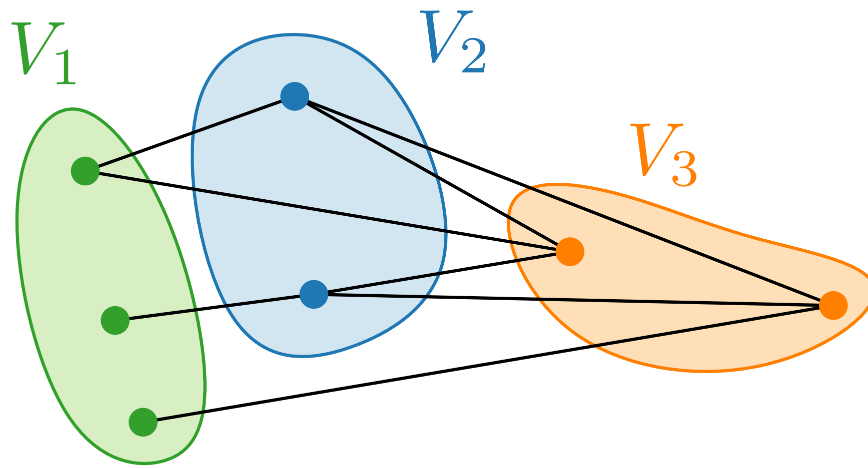
$\kappa = \mathbf{2} + \mathbf{1} + \mathbf{6} = 9$





SLE Parameterized by κ is W[1]-hard

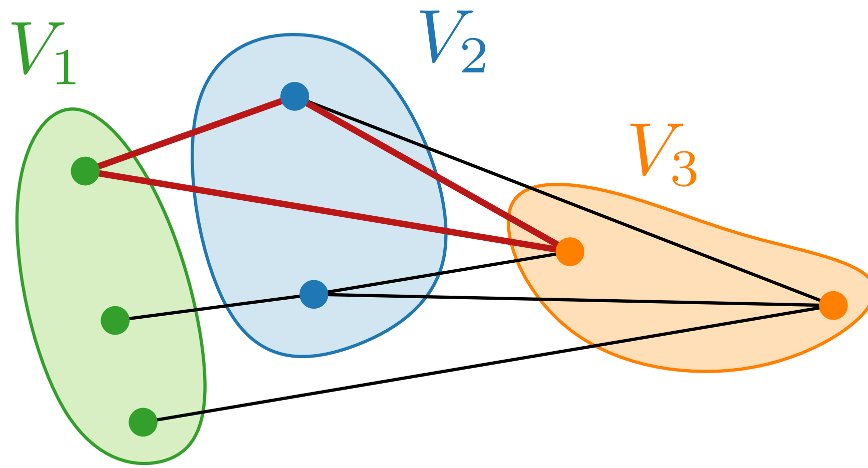
Reduction from Multi-Colored Clique (McC)



Parameter k = size of clique

SLE Parameterized by κ is W[1]-hard

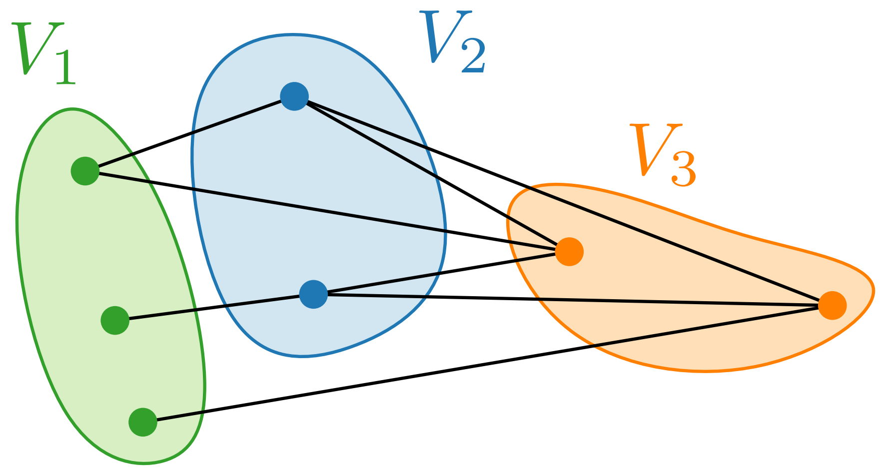
Reduction from Multi-Colored Clique (McC)

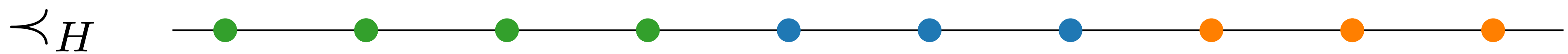


Parameter k = size of clique

SLE Parameterized by κ is W[1]-hard

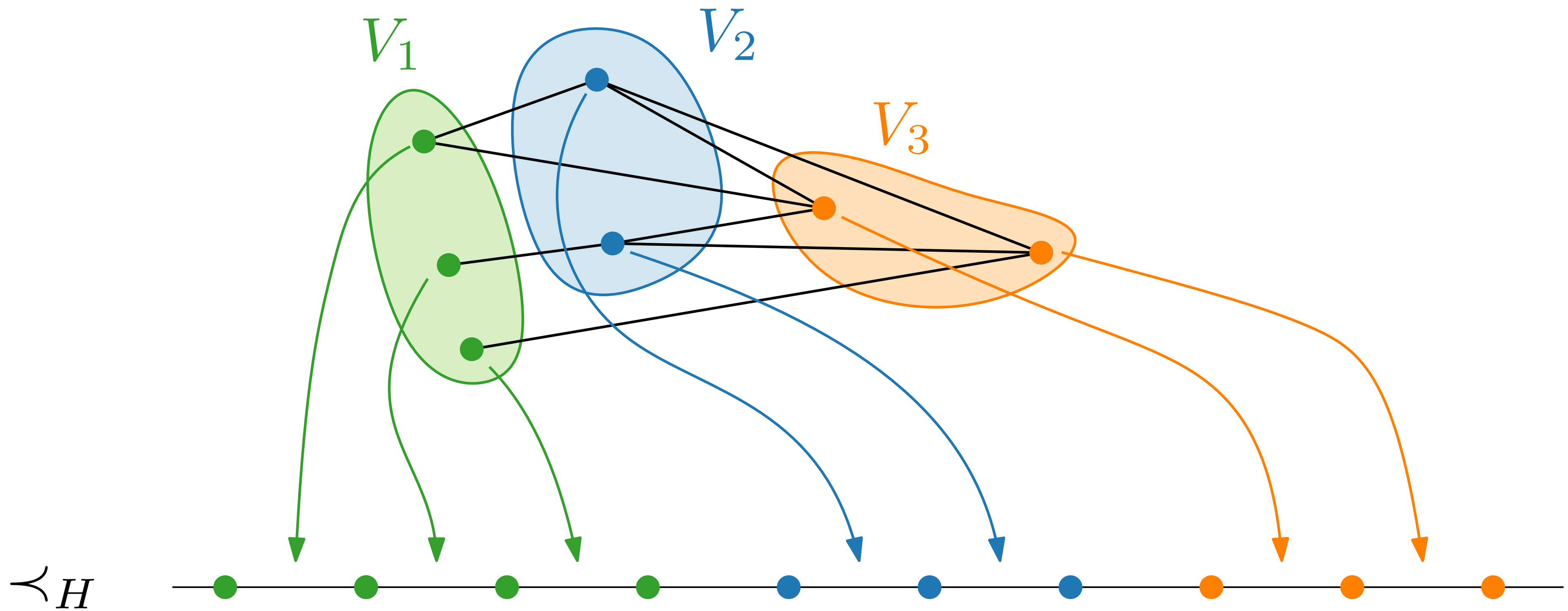
Reduction from Multi-Colored Clique (McC)





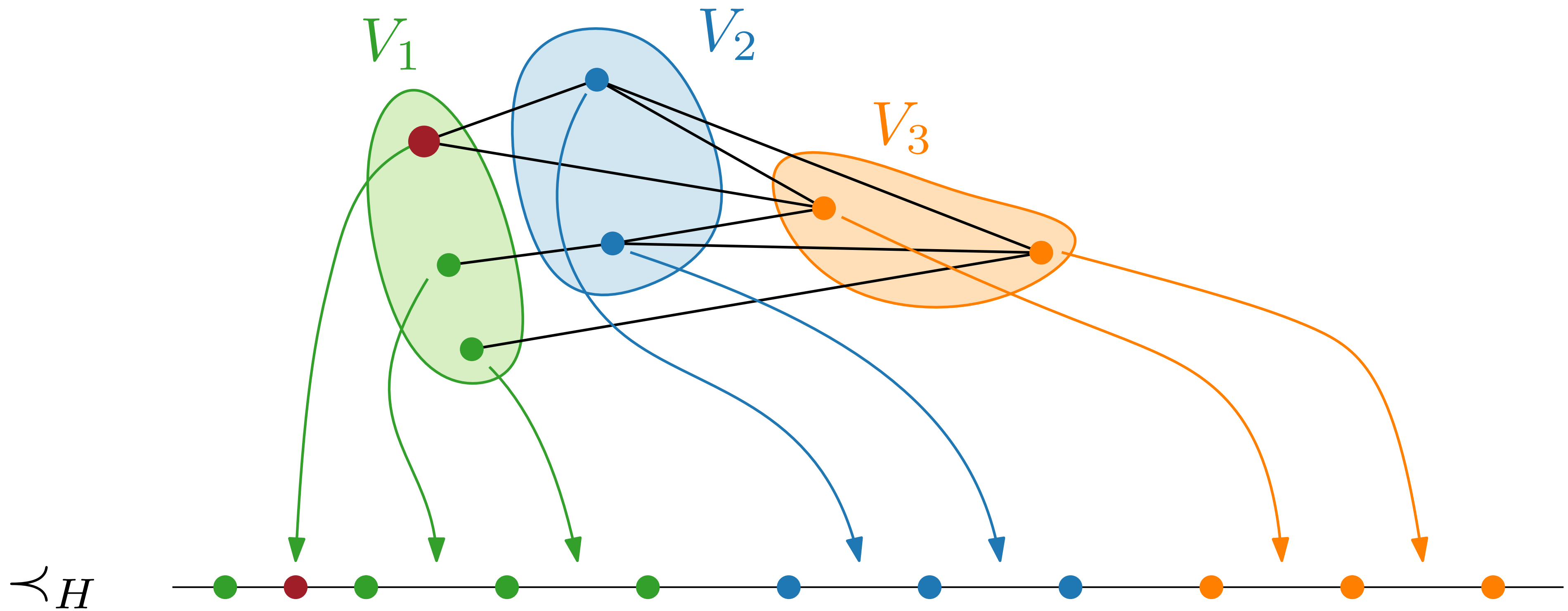
SLE Parameterized by κ is W[1]-hard

Reduction from Multi-Colored Clique (McC)



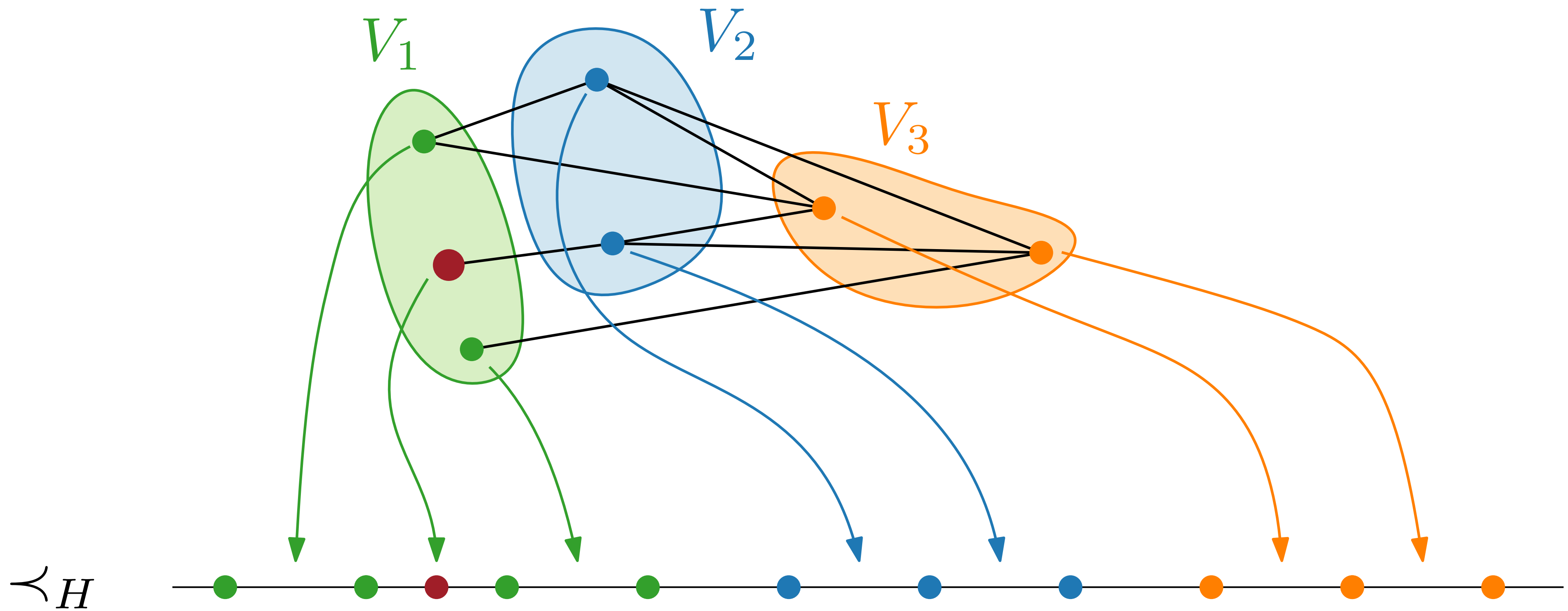
SLE Parameterized by κ is W[1]-hard

Reduction from Multi-Colored Clique (McC)



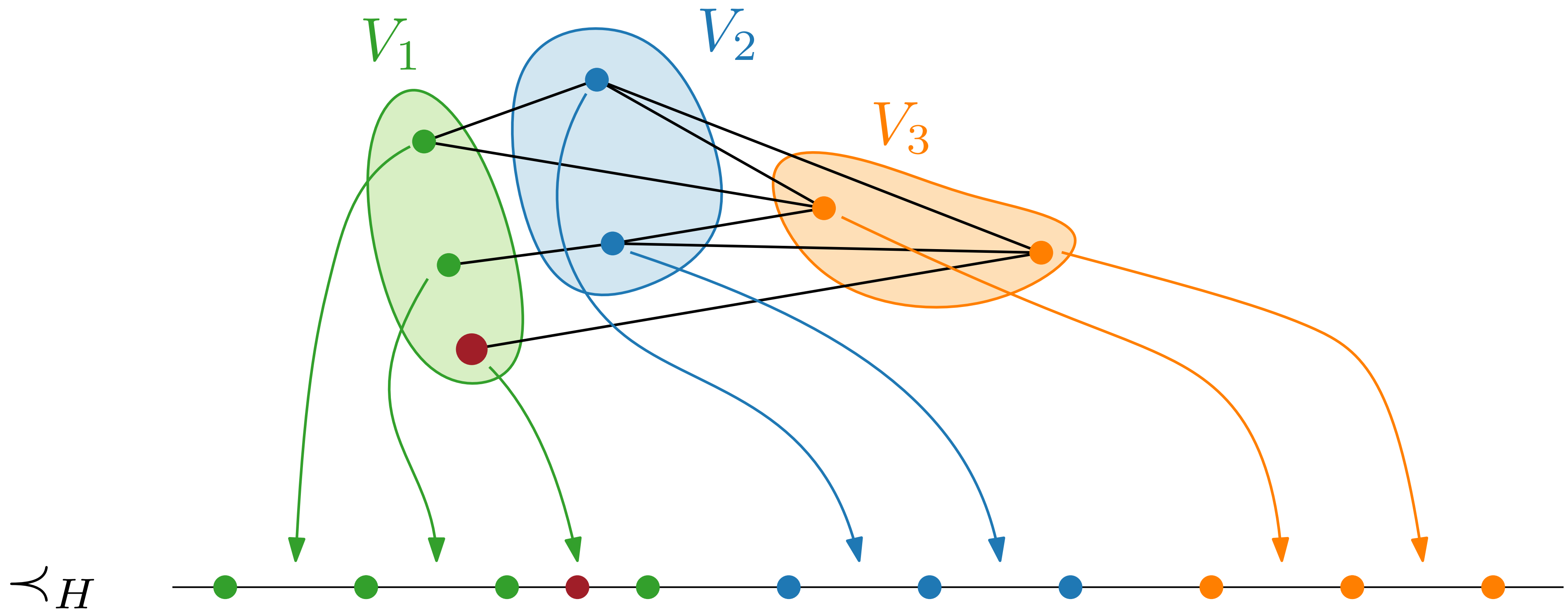
SLE Parameterized by κ is W[1]-hard

Reduction from Multi-Colored Clique (McC)



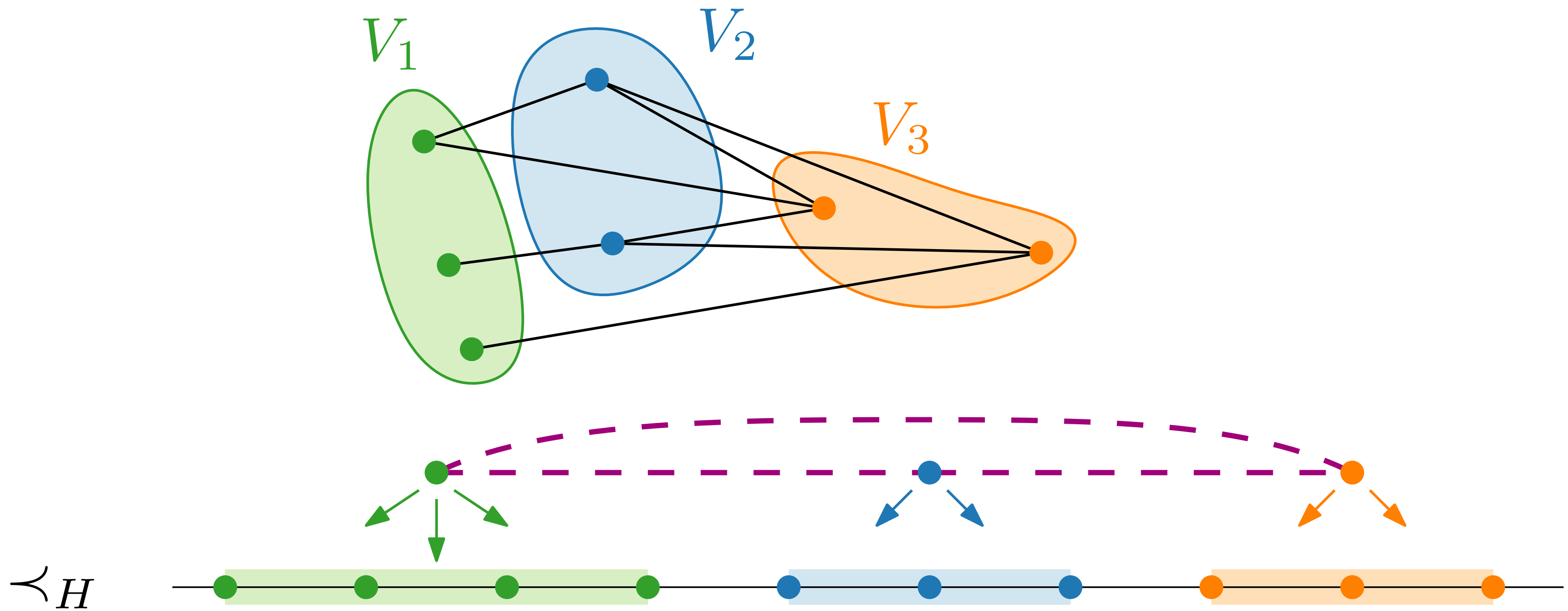
SLE Parameterized by κ is W[1]-hard

Reduction from Multi-Colored Clique (McC)



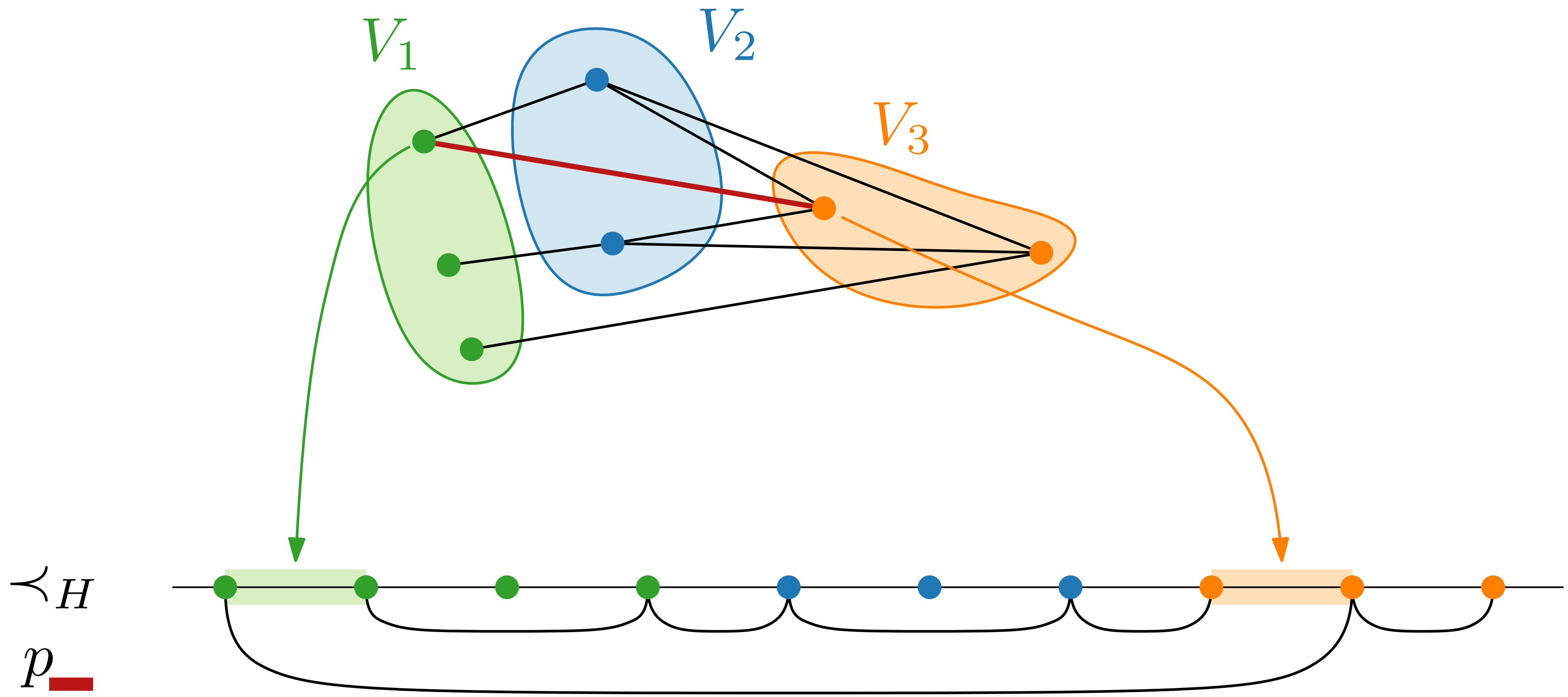
SLE Parameterized by κ is W[1]-hard

Reduction from MULTI-COLORED CLIQUE (McC)



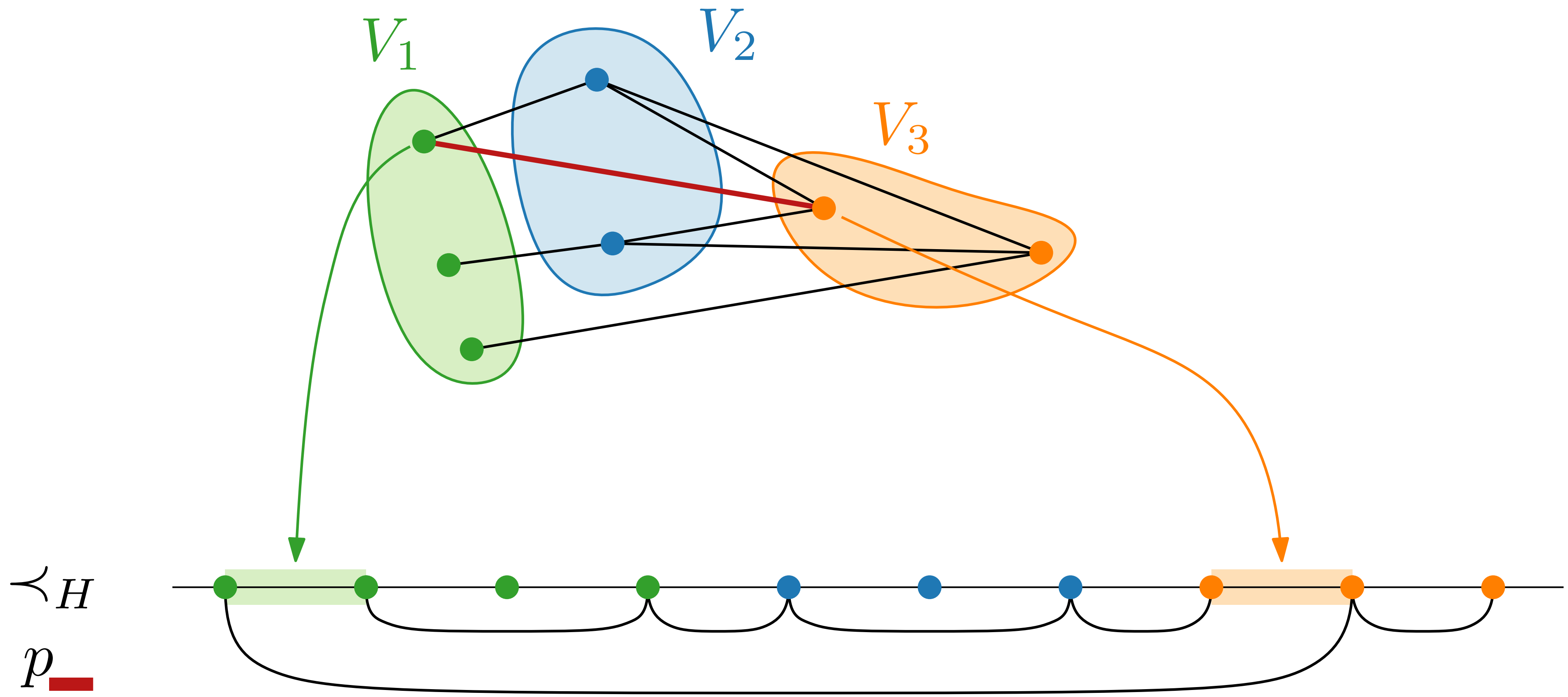
SLE Parameterized by κ is W[1]-hard

Reduction from Multi-Colored Clique (McC)



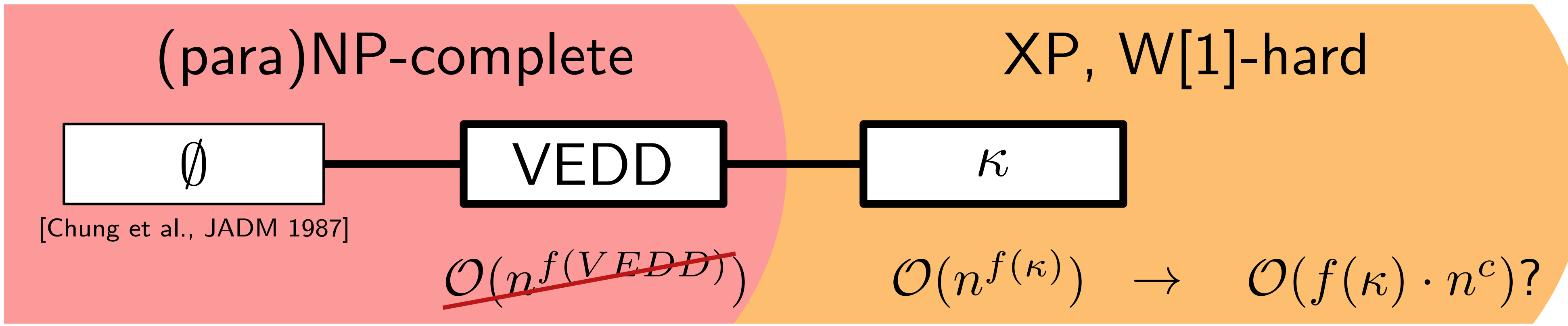
SLE Parameterized by κ is W[1]-hard

Reduction from MULTI-COLORED CLIQUE (MCC)



Theorem:
SLE parameterized by κ, i.e., the number of missing vertices and edges, is W[1]-hard.

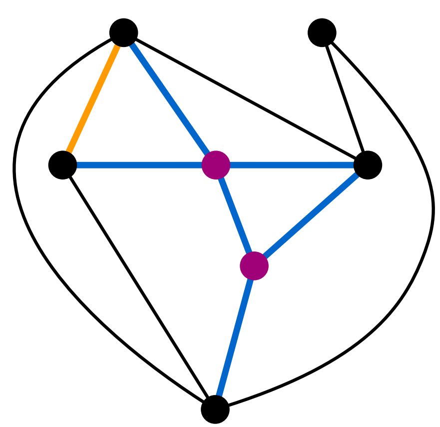
Our Results

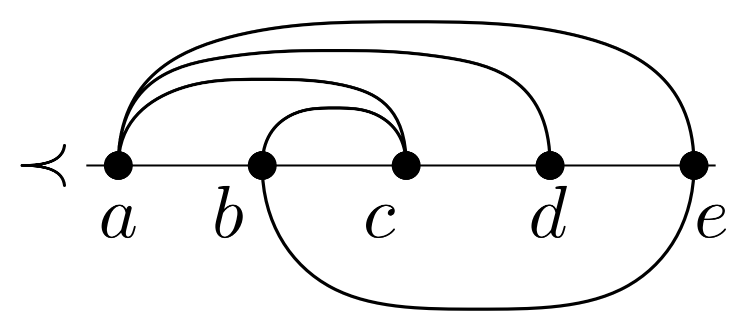


VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H
$\boldsymbol{\kappa}$**:** #missing vertices **plus** #missing edges

VEDD $= 2 + 1 = 3$

$\kappa = 2 + 1 + 6 = 9$





Our Results

(para)NP-complete | XP, W[1]-hard

$\emptyset$ — VEDD — κ

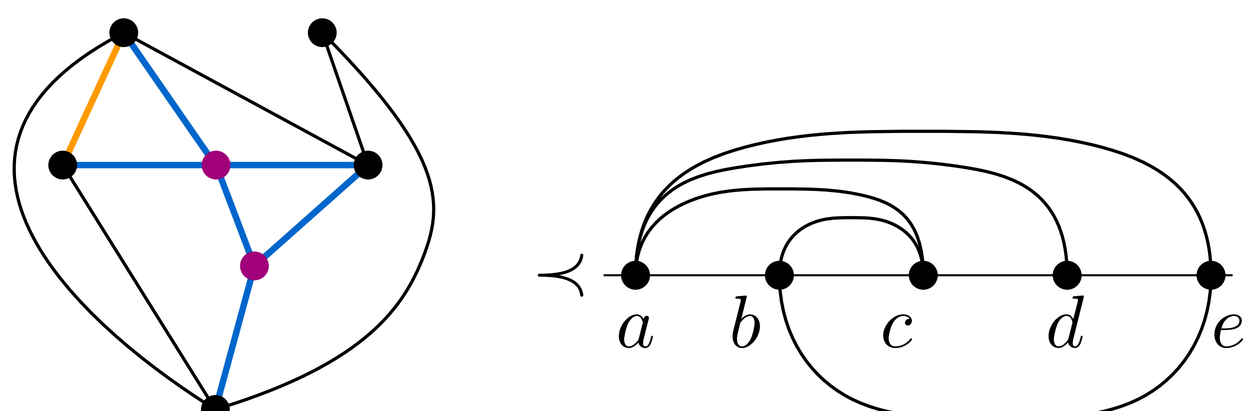
[Chung et al., JADM 1987]

~~$\mathcal{O}(n^{f(VEDD)})$~~ $\mathcal{O}(n^{f(\kappa)}) \rightarrow$ ~~$\mathcal{O}(f(\kappa) \cdot n^{c})$~~?

VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H
κ: #missing vertices **plus** #missing edges

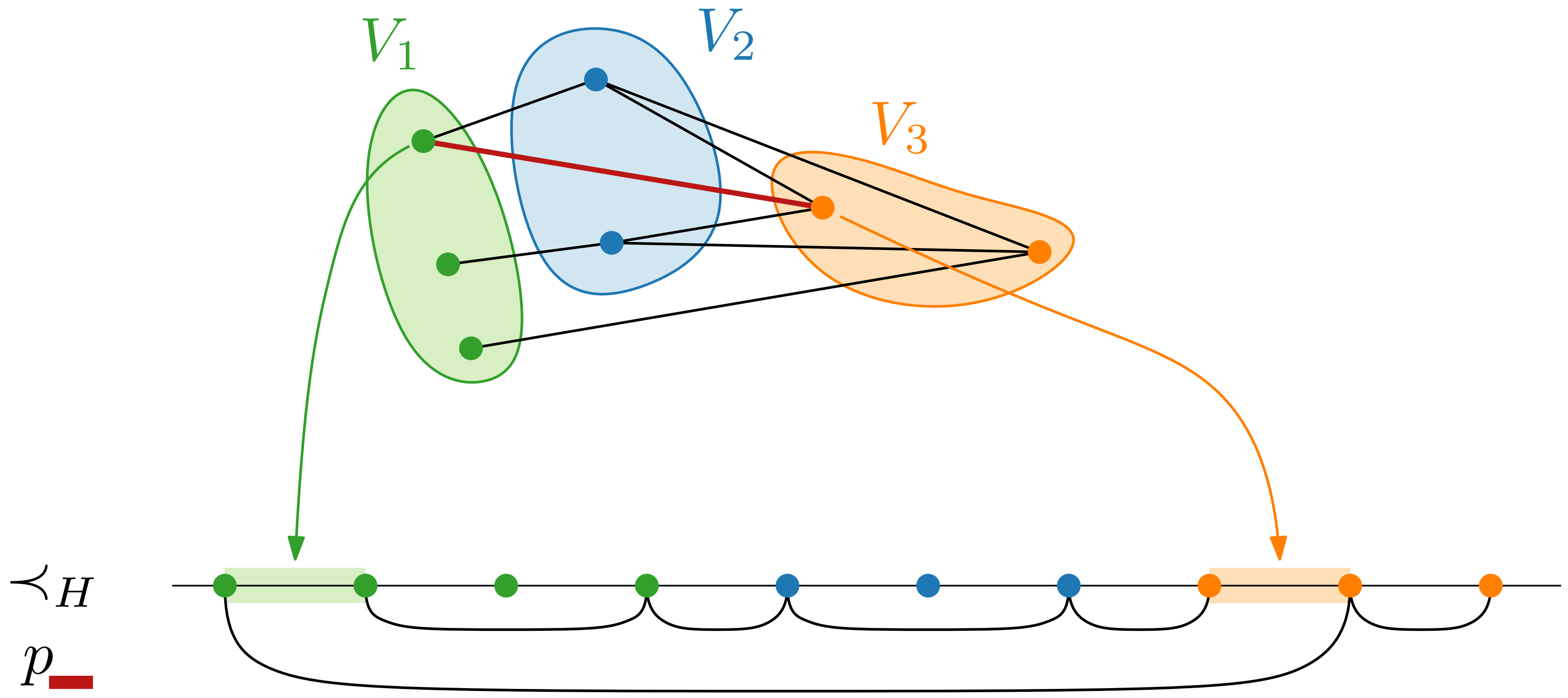
VEDD $= 2 + 1 = 3$

$\kappa = 2 + 1 + 6 = 9$



Rewind: SLE Parameterized by κ is W[1]-hard

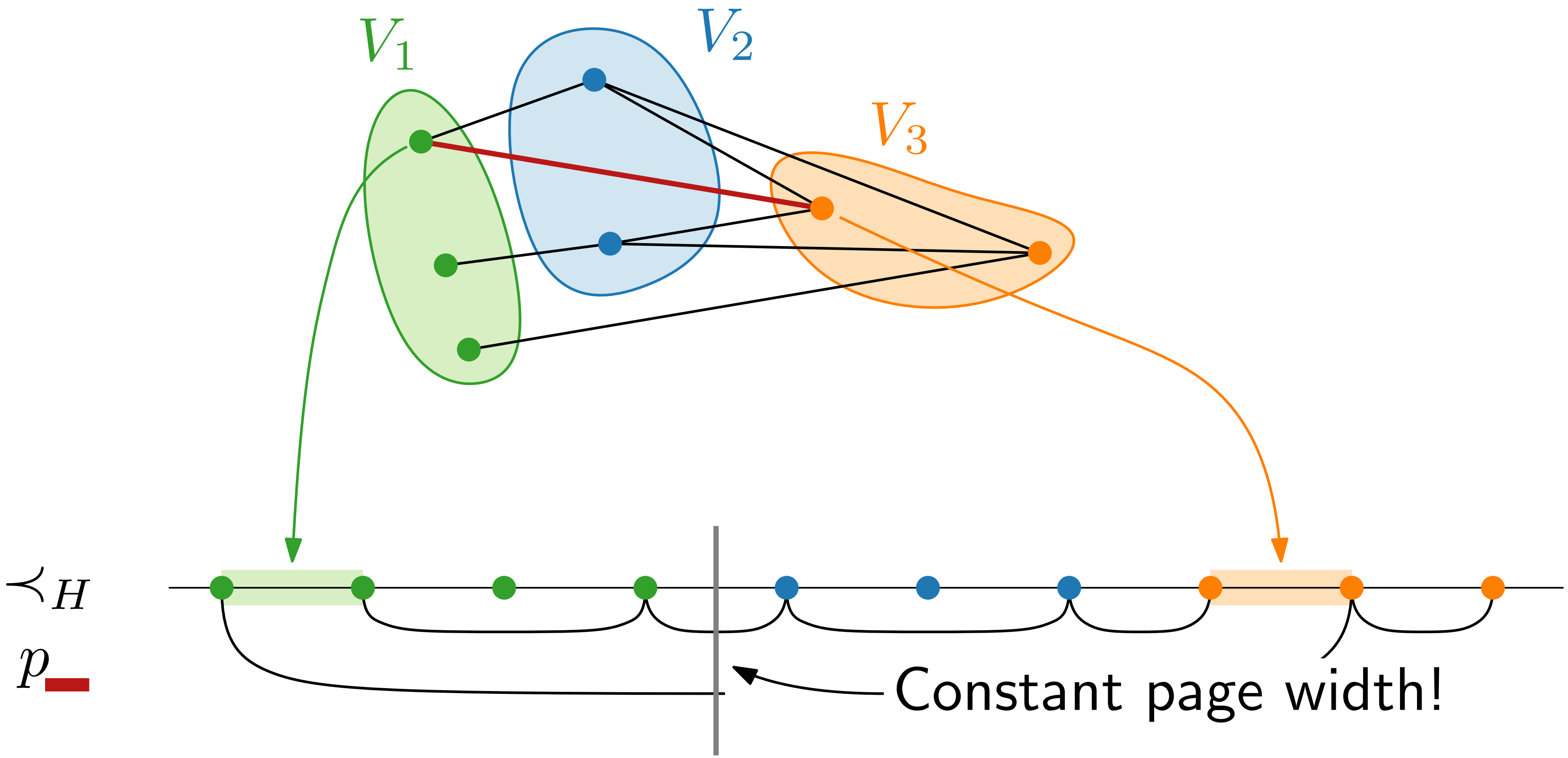
Reduction from MULTI-COLORED CLIQUE (MCC)



Theorem:
SLE parameterized by κ, i.e., the number of missing vertices and edges, is W[1]-hard.

Rewind: SLE Parameterized by κ is W[1]-hard

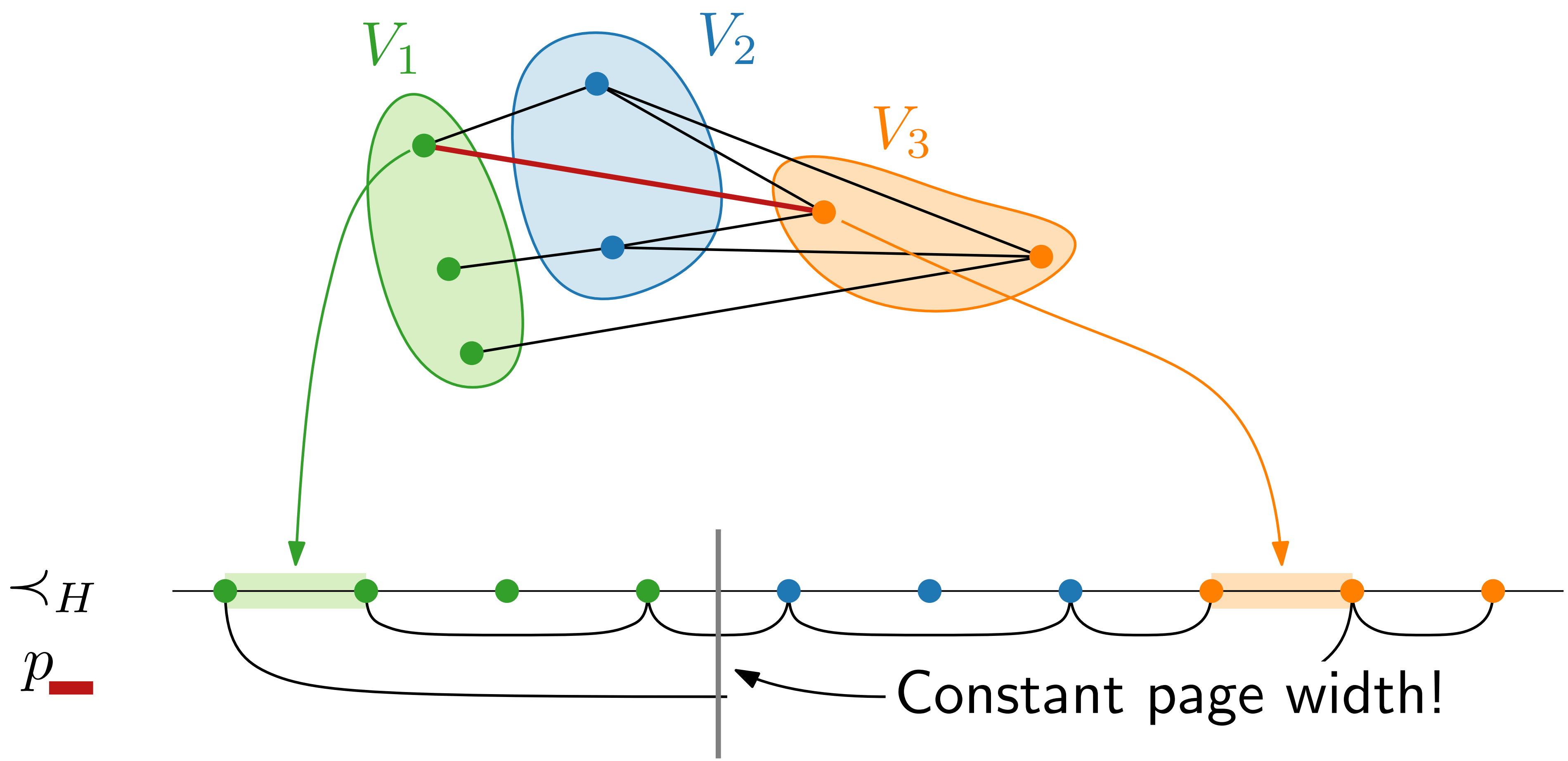
Reduction from MULTI-COLORED CLIQUE (MCC)



Theorem:
SLE parameterized by κ, i.e., the number of missing vertices and edges, is W[1]-hard.

Rewind: SLE Parameterized by κ is W[1]-hard

Reduction from Multi-Colored Clique (McC)



Theorem:
SLE parameterized by κ, i.e., the number of missing vertices and edges, and the **page width ω of the given layout** is W[1]-hard.

Our Results

(para)NP-complete | XP, W[1]-hard

$\emptyset$ — VEDD — κ

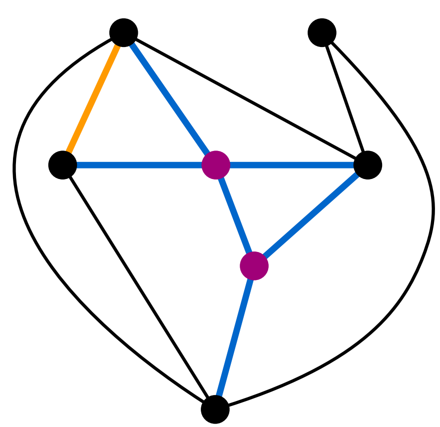
[Chung et al., JADM 1987]

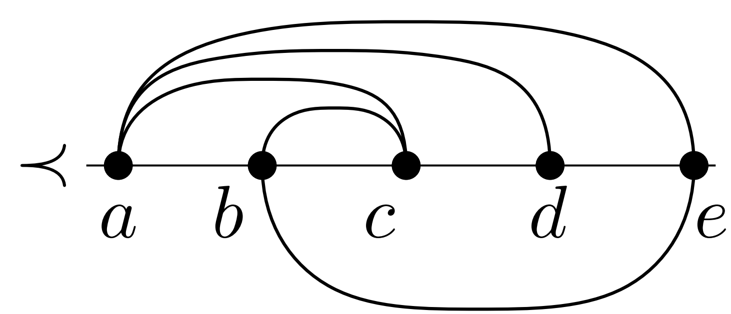
~~$\mathcal{O}(n^{f(VEDD)})$~~ $\mathcal{O}(n^{f(\kappa)}) \rightarrow$ ~~$\mathcal{O}(f(\kappa) \cdot n^c)$~~?

VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H
$\boldsymbol{\kappa}$**:** #missing vertices **plus** #missing edges

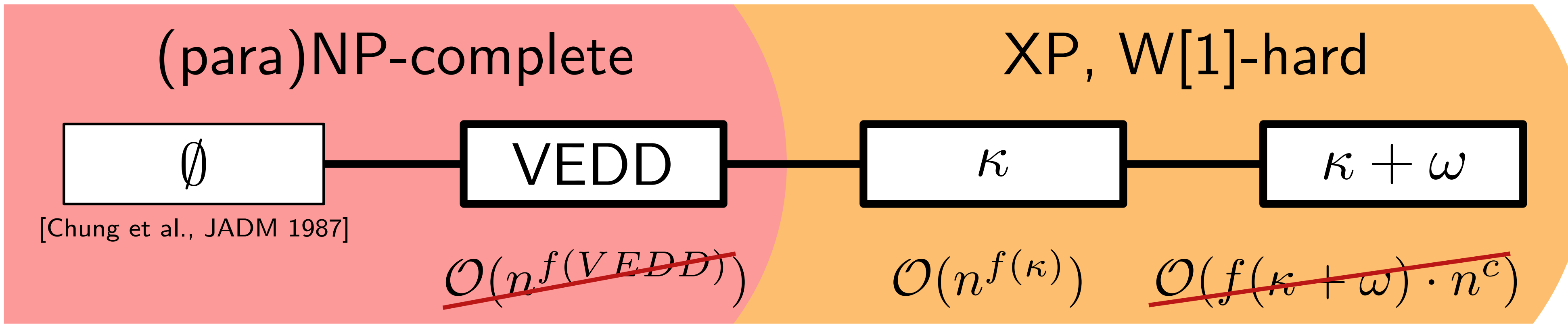
VEDD $=$ **2** $+$ **1** $= 3$

$\kappa =$ **2** $+$ **1** $+$ **6** $= 9$





Our Results



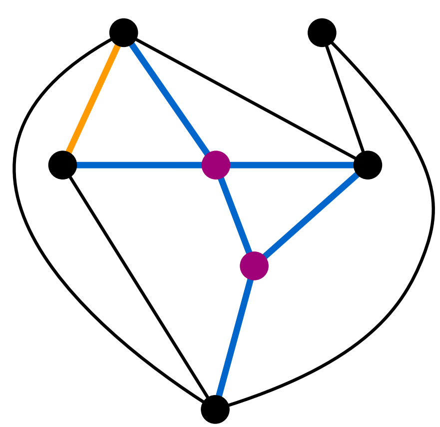
VEDD: #vertices (inc. **incident edges**) & edges to delete from G to obtain H

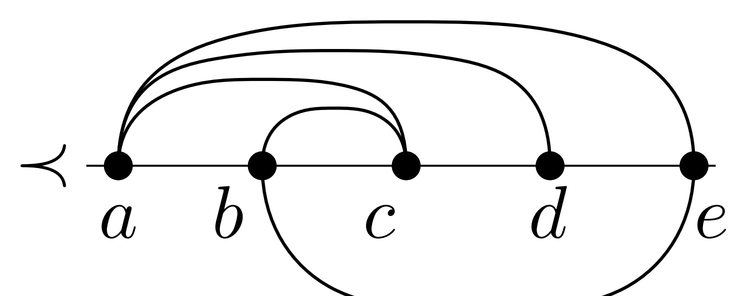
$\boldsymbol{\kappa}$**:** #missing vertices **plus** #missing edges

$\boldsymbol{\omega}$**:** page width of $\langle \prec_H, \sigma_H \rangle$

VEDD = **2** + **1** = 3

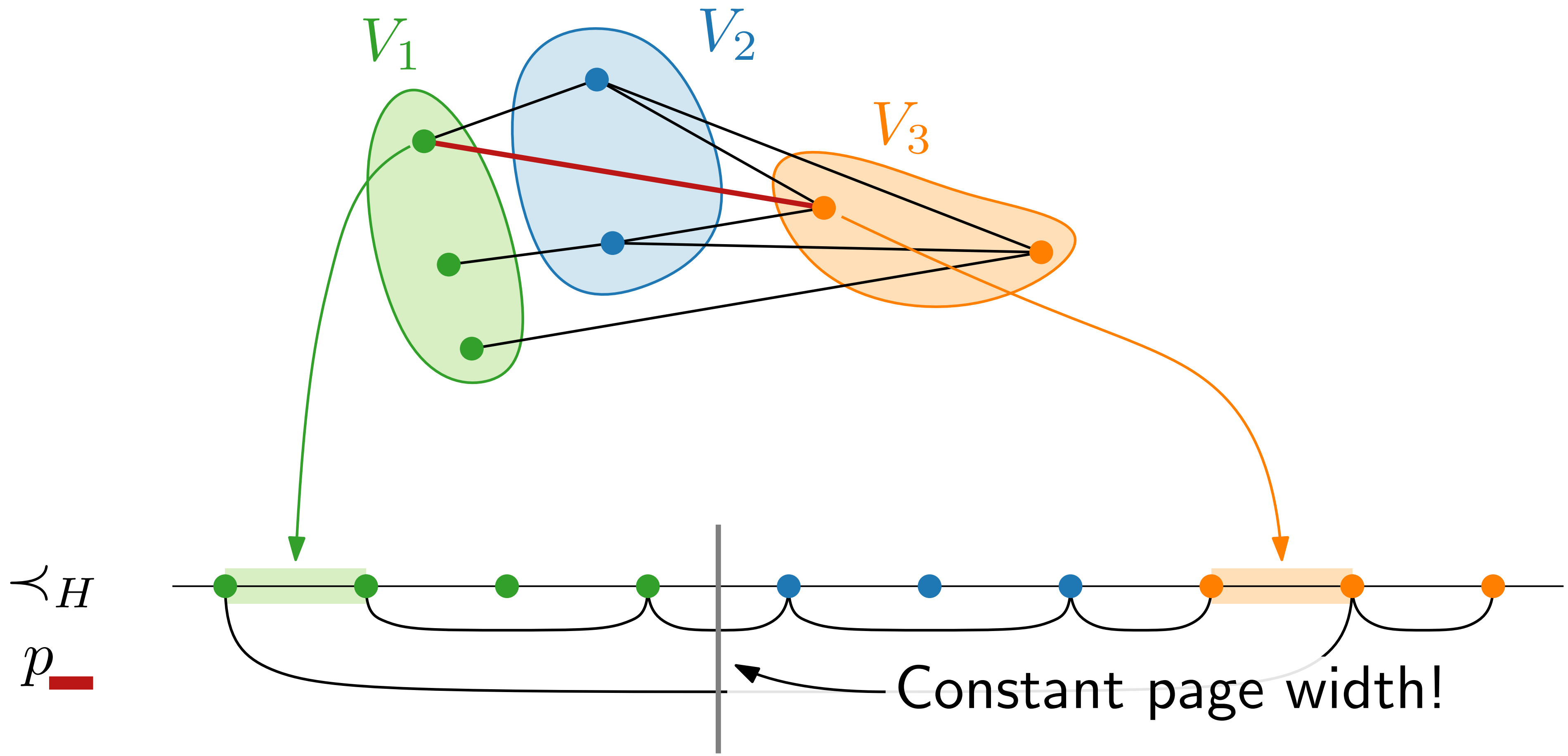
κ = **2** + **1** + **6** = 9





Rewind II: SLE Parameterized by κ is W[1]-hard

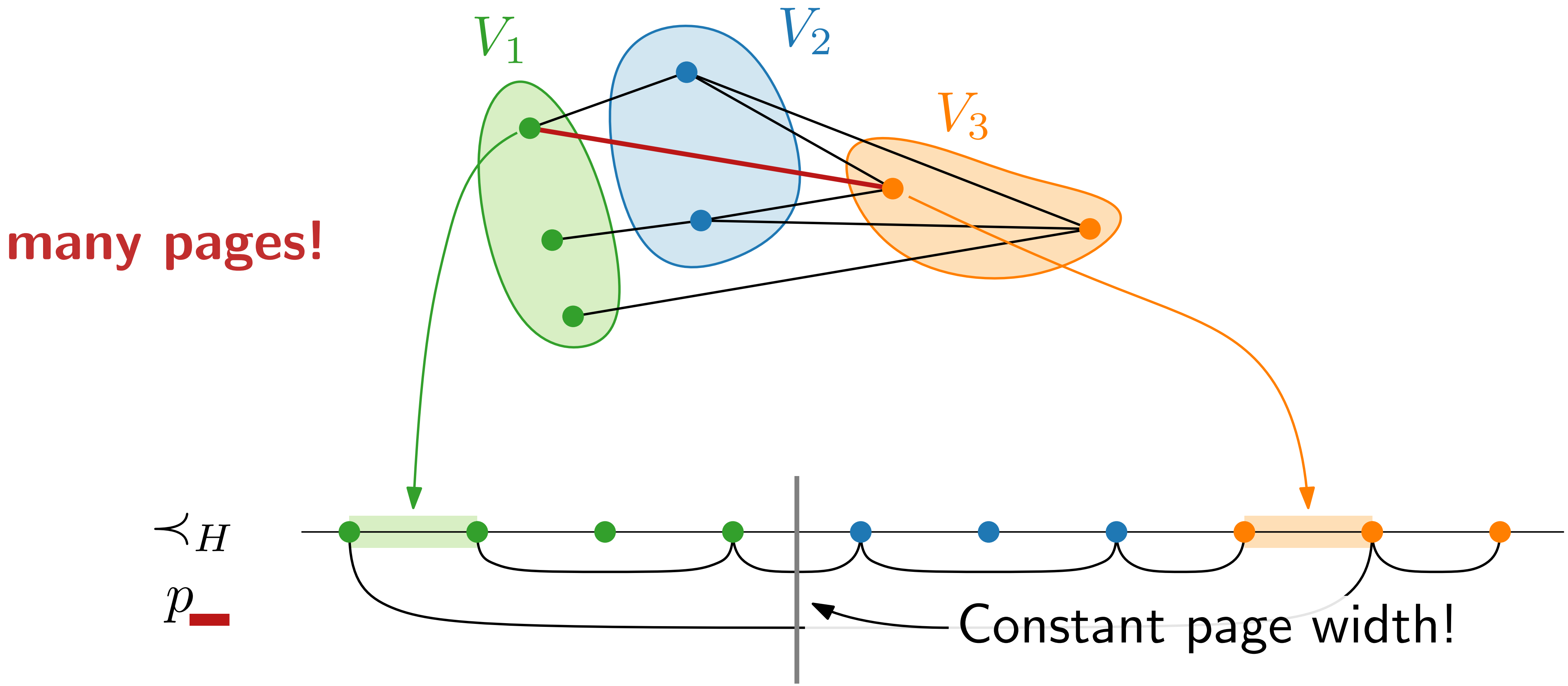
Reduction from Multi-Colored Clique (McC)



Theorem:
SLE parameterized by κ, i.e., the number of missing vertices and edges, and the **page width ω of the given layout** is W[1]-hard.

Rewind II: SLE Parameterized by κ is W[1]-hard

Reduction from MULTI-COLORED CLIQUE (MCC)



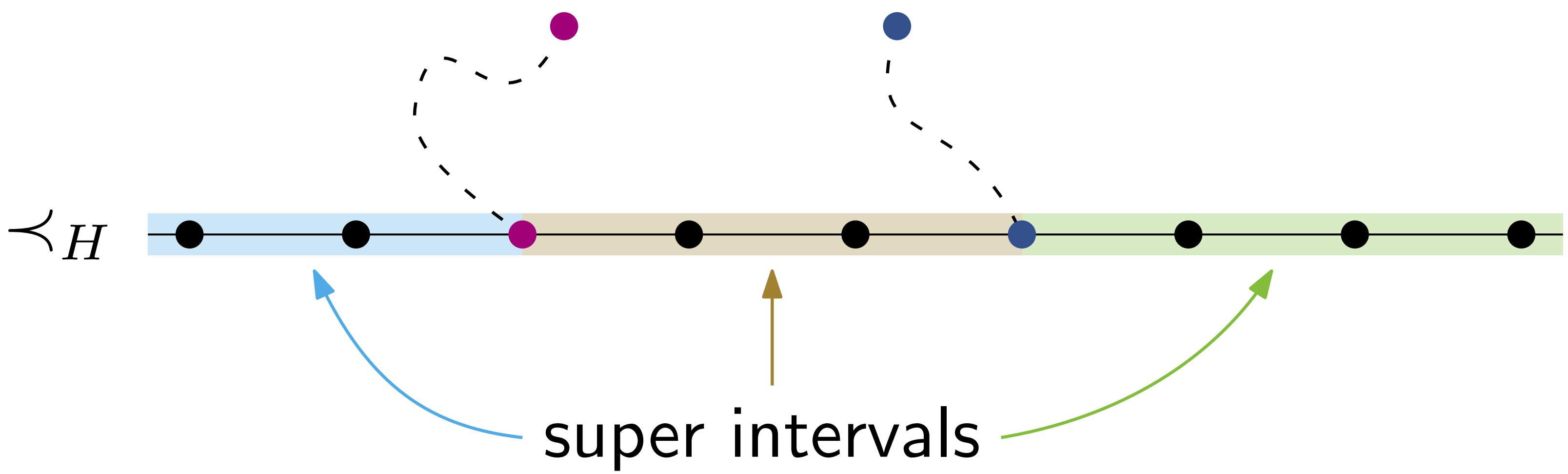
Theorem:
SLE parameterized by κ, i.e., the number of missing vertices and edges, and the **page width ω of the given layout** is W[1]-hard.

An FPT-Algorithm Parameterized by κ, ω, and ℓ

Step 1: Guess the final page assignment σ_G

Step 2: Guess the relative order among missing vertices

Step 3: Guess the assignment of missing vertices to *super-intervals*

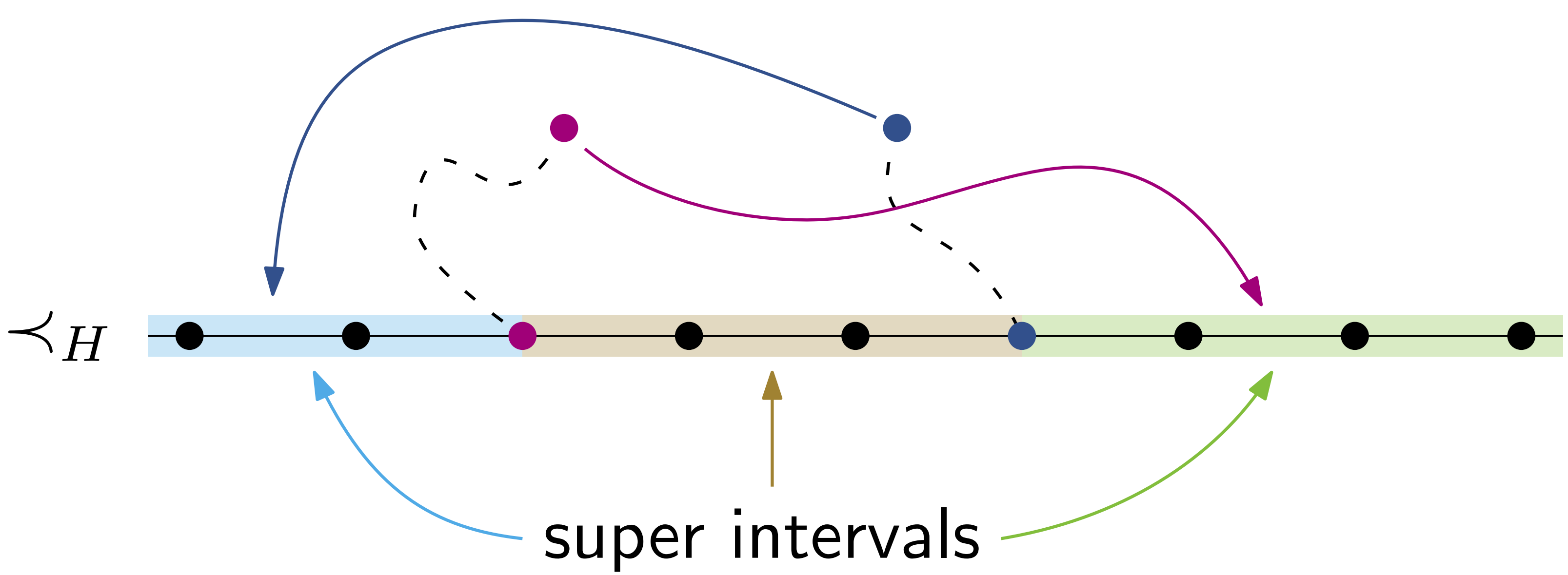


An FPT-Algorithm Parameterized by κ, ω, and ℓ

Step 1: Guess the final page assignment σ_G

Step 2: Guess the relative order among missing vertices

Step 3: Guess the assignment of missing vertices to *super-intervals*



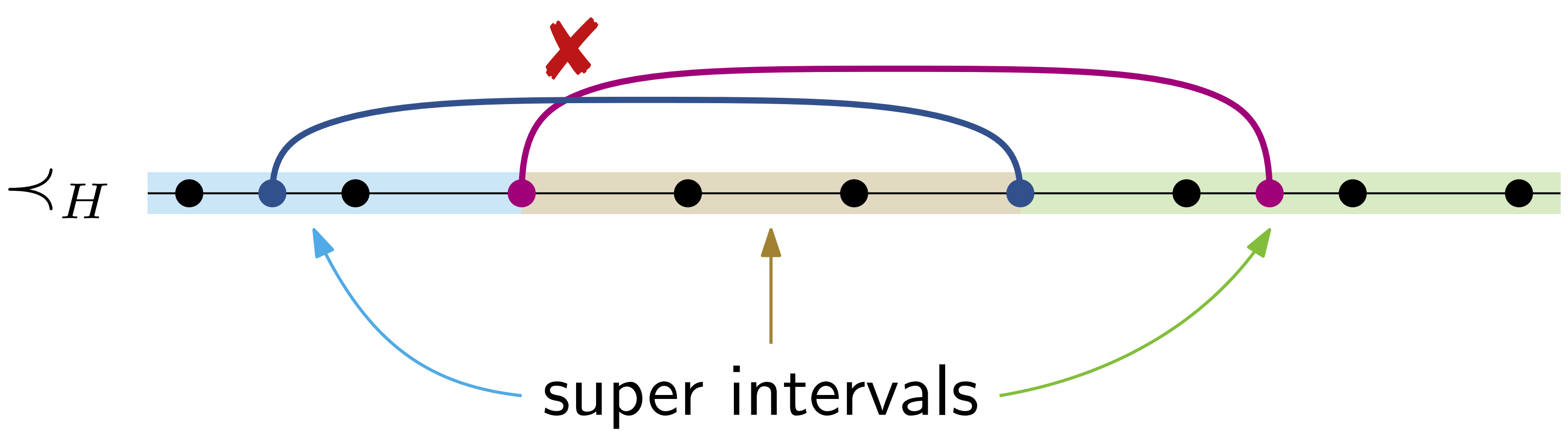
An FPT-Algorithm Parameterized by κ, ω, and ℓ

Step 1: Guess the final page assignment σ_G

Step 2: Guess the relative order among missing vertices

Step 3: Guess the assignment of missing vertices to *super-intervals*

$\Rightarrow$ Steps 1 – 3: $\mathcal{O}(f(\kappa + \ell))$ branches



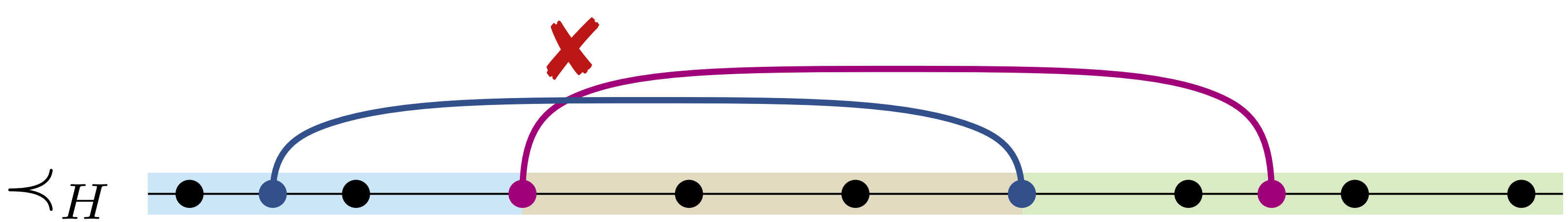
An FPT-Algorithm Parameterized by κ, ω, and ℓ

Step 1: Guess the final page assignment σ_G

Step 2: Guess the relative order among missing vertices

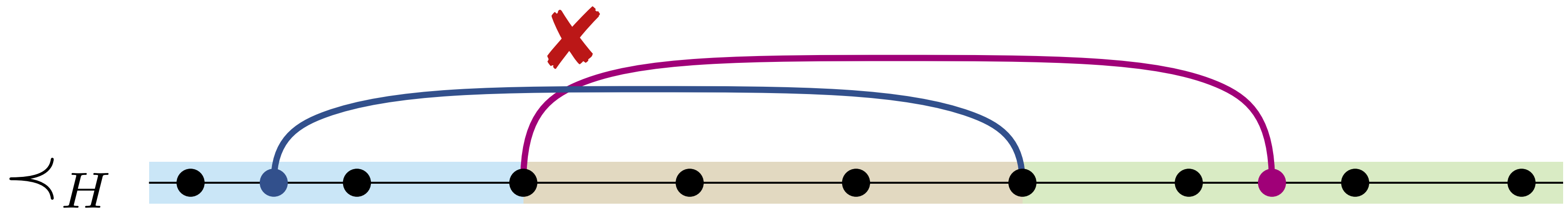
Step 3: Guess the assignment of missing vertices to *super-intervals*

$\Rightarrow$ Steps 1 – 3: $\mathcal{O}(f(\kappa + \ell))$ branches

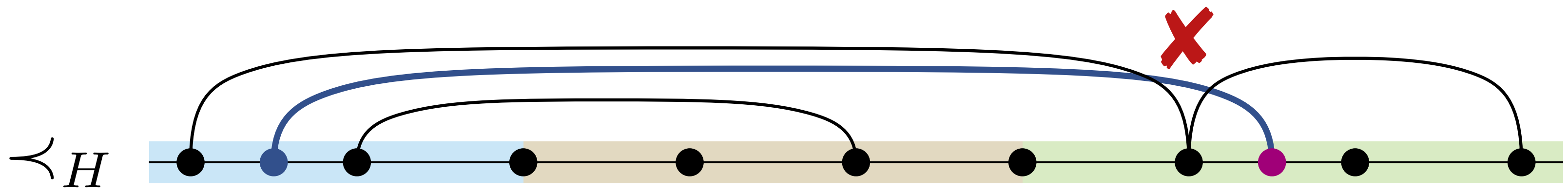


So are we done?

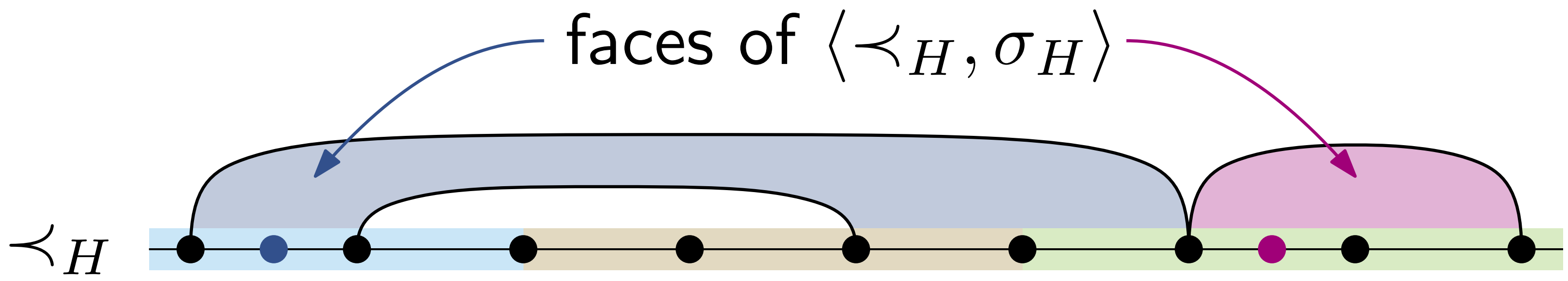
A Dynamic Program To Prevent Remaining Crossings



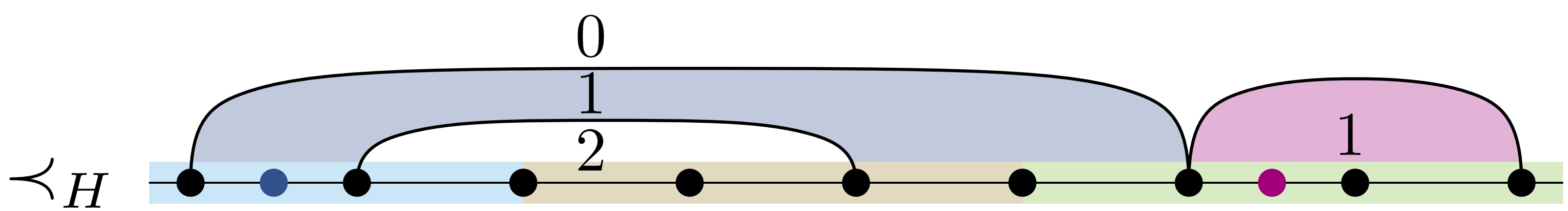
A Dynamic Program To Prevent Remaining Crossings



A Dynamic Program To Prevent Remaining Crossings

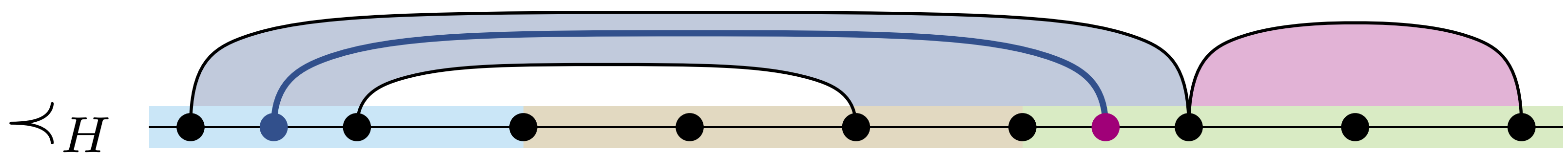


A Dynamic Program To Prevent Remaining Crossings



Step 4: Guess for each missing edge its distance to the outer face

A Dynamic Program To Prevent Remaining Crossings



Step 4: Guess for each missing edge its distance to the outer face

Step 5: Apply a DP in each branch to check if extended stack layout exists

$\mathcal{O}(n_{\mathsf{add}} m_{\mathsf{add}} |\mathcal{I}|)$ time per branch

Finally FPT!

Step 1: Guess the final page assignment σ_G

Step 2: Guess the relative order among missing vertices

Step 3: Guess the assignment of missing vertices to super-intervals

Step 4: Guess for each missing edge its distance to the outer face

Step 5: Apply a DP in each branch to check if extended stack layout exists

Finally FPT!

Step 1: Guess the final page assignment σ_G

Step 2: Guess the relative order among missing vertices

Step 3: Guess the assignment of missing vertices to super-intervals

Step 4: Guess for each missing edge its distance to the outer face

Step 5: Apply a DP in each branch to check if extended stack layout exists

Theorem:
$\textsc{SLE}$ can be solved in $\mathcal{O}(\ell^{m_{\mathsf{add}}} \cdot n_{\mathsf{add}}! \cdot {m_{\mathsf{add}}}^{n_{\mathsf{add}}} \cdot \omega^{m_{\mathsf{add}}} \cdot n_{\mathsf{add}} m_{\mathsf{add}} |\mathcal{I}|)$ time.

Finally FPT!

Step 1: Guess the final page assignment σ_G

Step 2: Guess the relative order among missing vertices

Step 3: Guess the assignment of missing vertices to super-intervals

Step 4: Guess for each missing edge its distance to the outer face

Step 5: Apply a DP in each branch to check if extended stack layout exists

#missing edges

Size of instance

#pages

#missing vertices

page width

Theorem:
SLE can be solved in $\mathcal{O}(\ell^{m_{\text{add}}} \cdot n_{\text{add}}! \cdot {m_{\text{add}}}^{n_{\text{add}}} \cdot \omega^{m_{\text{add}}} \cdot n_{\text{add}} m_{\text{add}} \, |\mathcal{I}|)$ time.

Finally FPT!

Step 1: Guess the final page assignment σ_G

Step 2: Guess the relative order among missing vertices

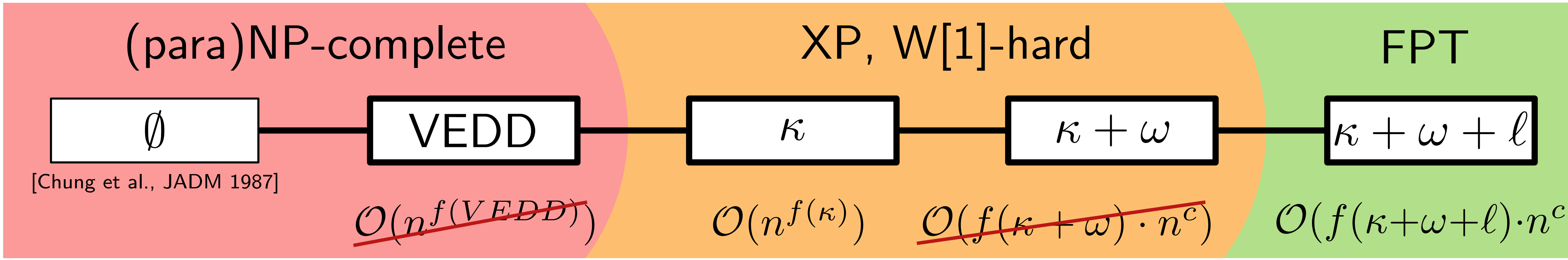
Step 3: Guess the assignment of missing vertices to super-intervals

Step 4: Guess for each missing edge its distance to the outer face

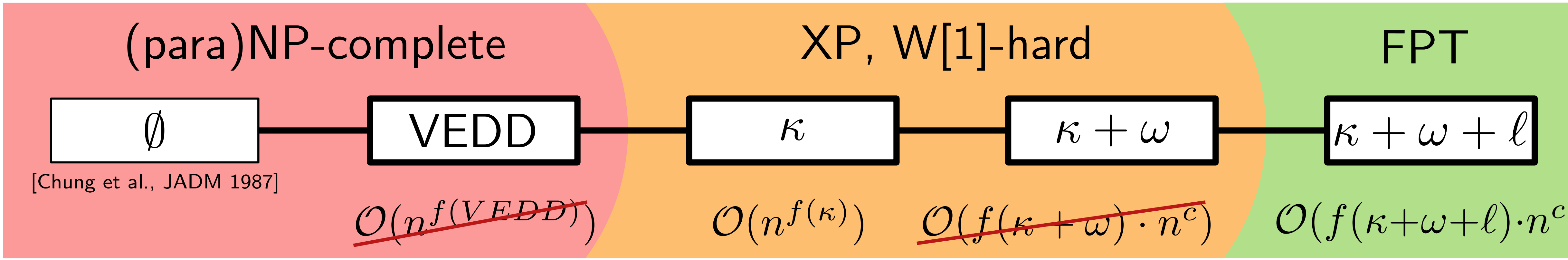
Step 5: Apply a DP in each branch to check if extended stack layout exists

Theorem:
SLE can be solved in $\mathcal{O}(f(\kappa + \omega + \ell) |\mathcal{I}|)$ time.

Conclusion


(para)NP-complete
XP, W[1]-hard
FPT
$\emptyset$
VEDD
κ
$\kappa + \omega$
$\kappa + \omega + \ell$
[Chung et al., JADM 1987]
$\mathcal{O}(n^{f(VEDD)})$
$\mathcal{O}(n^{f(\kappa)})$
$\mathcal{O}(f(\kappa + \omega) \cdot n^c)$
$\mathcal{O}(f(\kappa+\omega+\ell) \cdot n^c)$

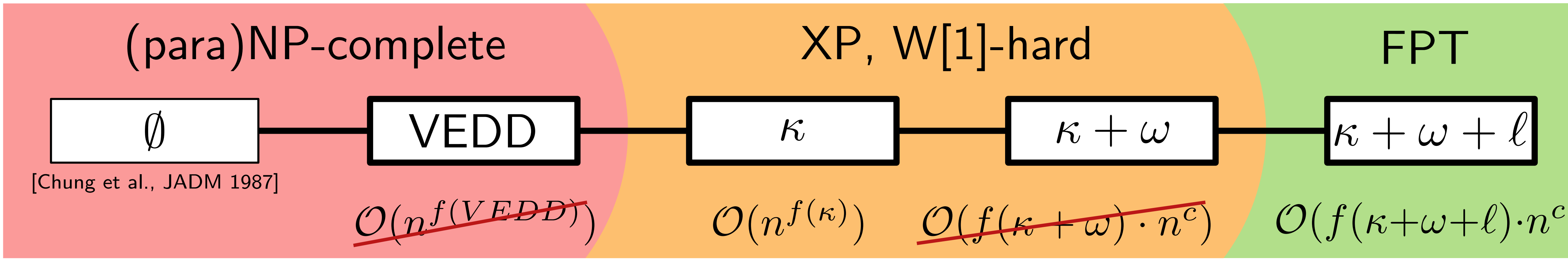
Conclusion



Future work:

Parameterization by $\kappa + \ell$: FPT or W[1]-hard?

Conclusion

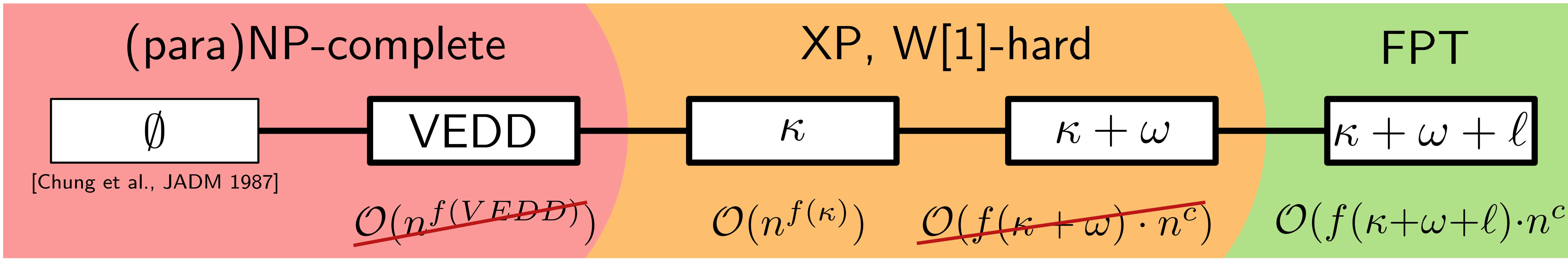


Future work:

Parameterization by $\kappa + \ell$: FPT or W[1]-hard?

Queue Layout Extension?

Conclusion



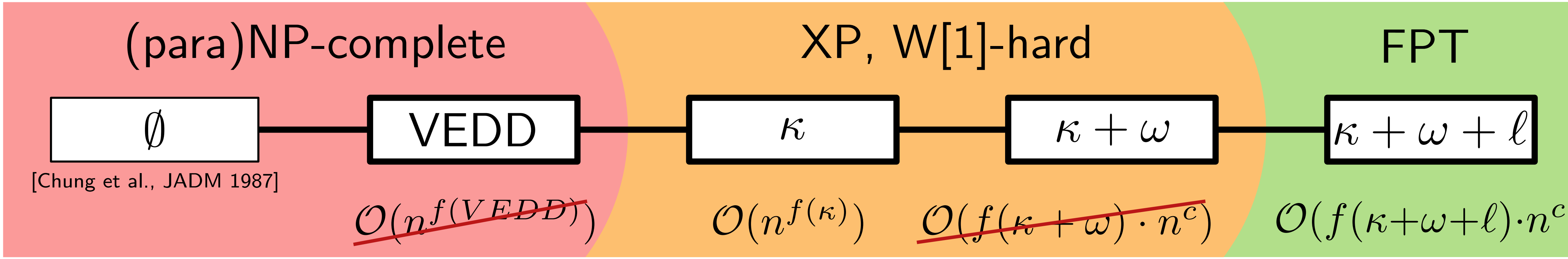
Future work:

Parameterization by $\kappa + \ell$: FPT or W[1]-hard?

Queue Layout Extension?

Generalized notion of extension: Given spine order for some vertices and page assignment for some edges, does there exist a stack layout that extends both?

Conclusion



Future work:

Parameterization by $\kappa + \ell$: FPT or W[1]-hard?

Queue Layout Extension?

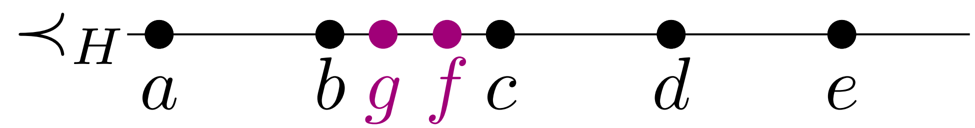
Generalized notion of extension: Given spine order for some vertices and page assignment for some edges, does there exist a stack layout that extends both?

Thank you for your attention!

App.: SLE Parameterized by κ is in XP

Step 1: Guess the extended spine order $\prec_G$

$\mathcal{O}(|\mathcal{I}|^{n_{add}})$ branches

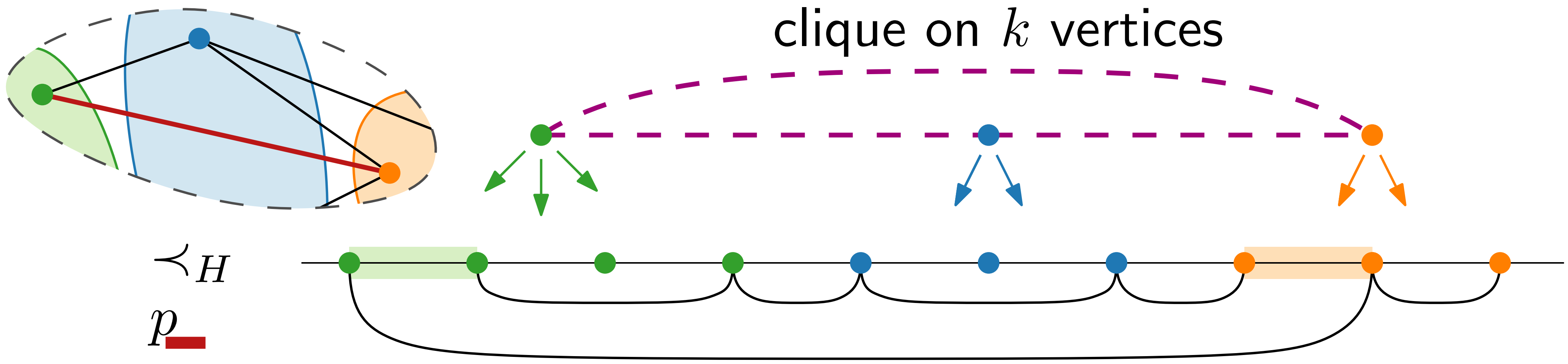


Step 2: Solve instance of SLE with only missing edges

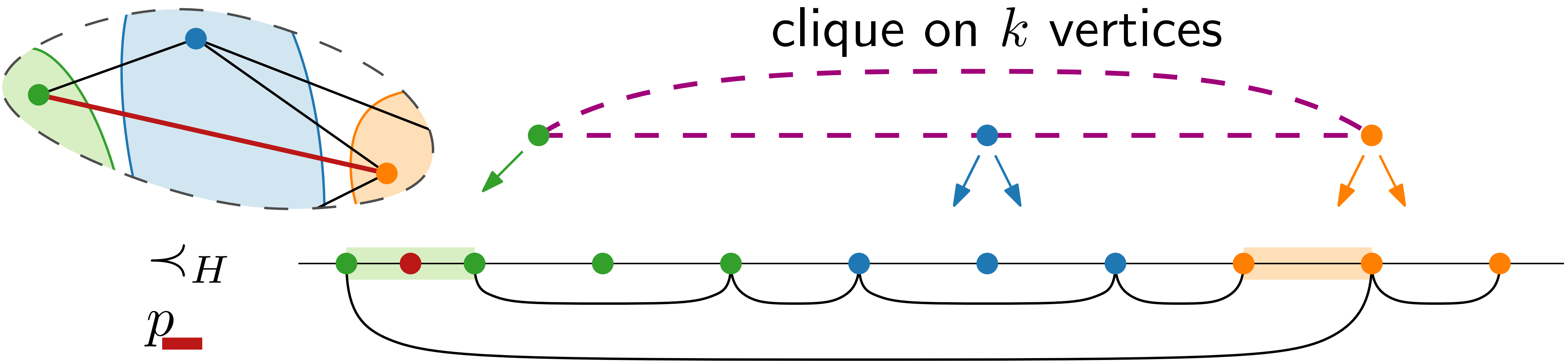
$\mathcal{O}({m_{add}}^{m_{add}} \cdot |\mathcal{I}|)$ time per branch

Theorem:
SLE can be solved in $\mathcal{O}(|\mathcal{I}|^{n_{\text{add}}+1} \cdot {m_{\text{add}}}^{m_{\text{add}}})$ time.

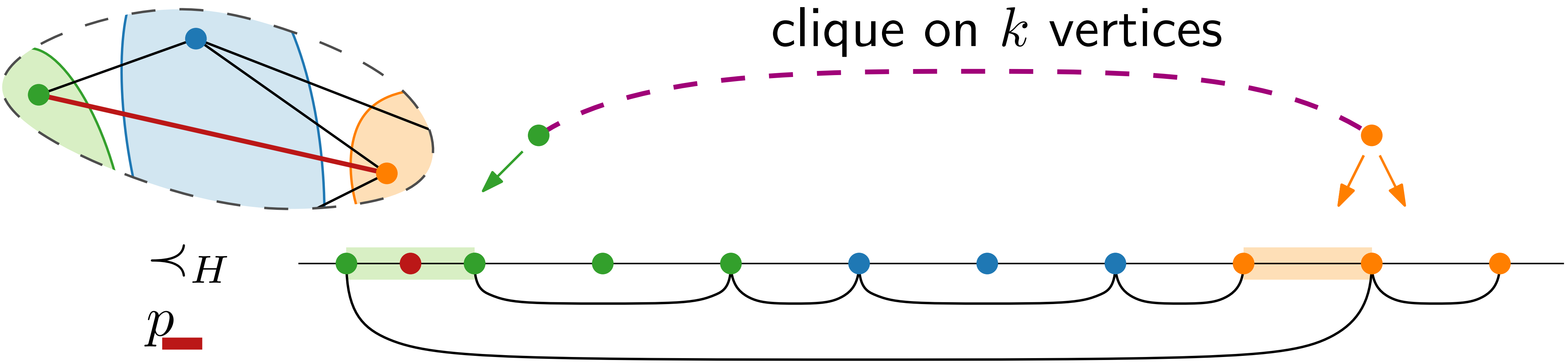
App.: SLE Parameterized by κ is W[1]-hard – Tunnel



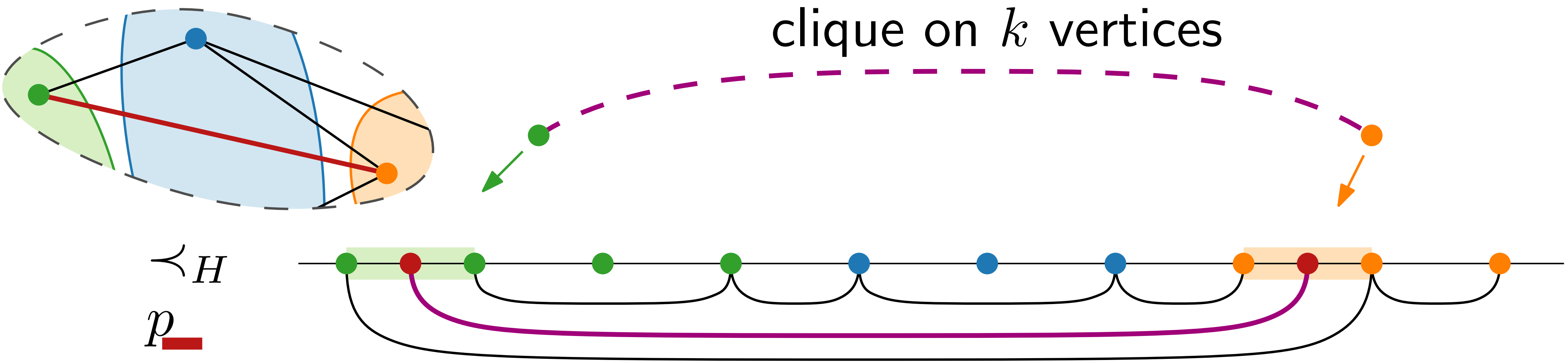
App.: SLE Parameterized by κ is W[1]-hard – Tunnel



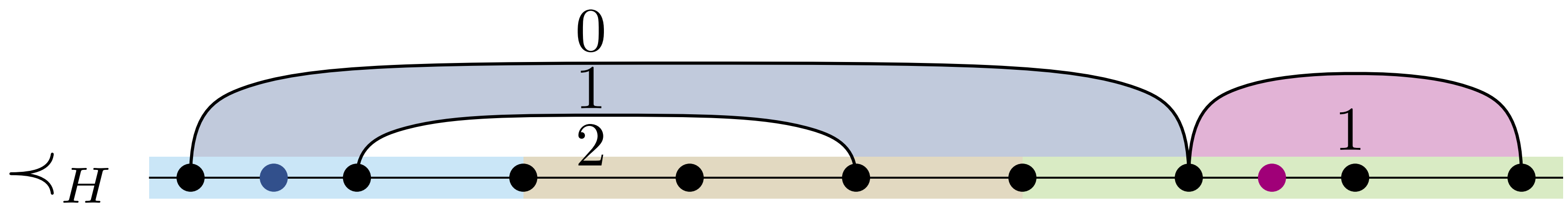
App.: SLE Parameterized by κ is W[1]-hard – Tunnel



App.: SLE Parameterized by κ is W[1]-hard – Tunnel



A Dynamic Program To Prevent the Remaining Crossings



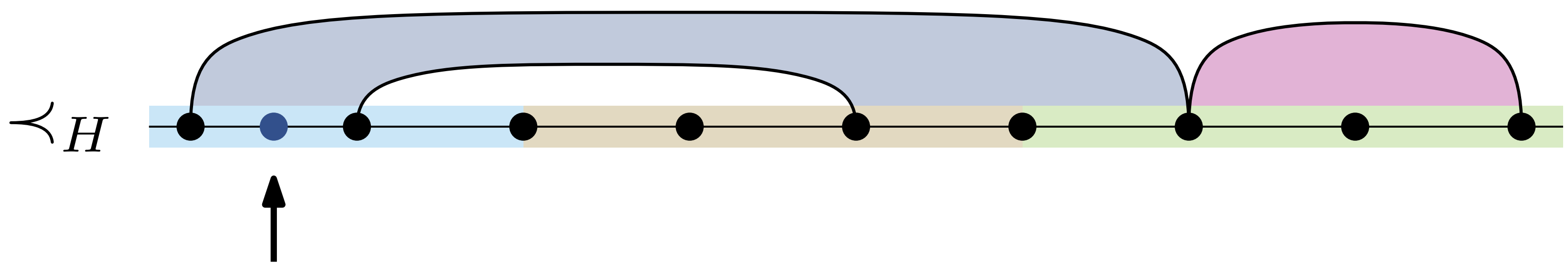
Number faces from outside-in $\Rightarrow$ numbers bounded by ω

Step 4: Guess for each new edge its distance to the outerface

Step 5: Apply a DP in each branch to compute stack layout

$\mathcal{O}(n_{\text{add}} m_{\text{add}} |\mathcal{I}|)$ time per branch

A Dynamic Program To Prevent the Remaining Crossings



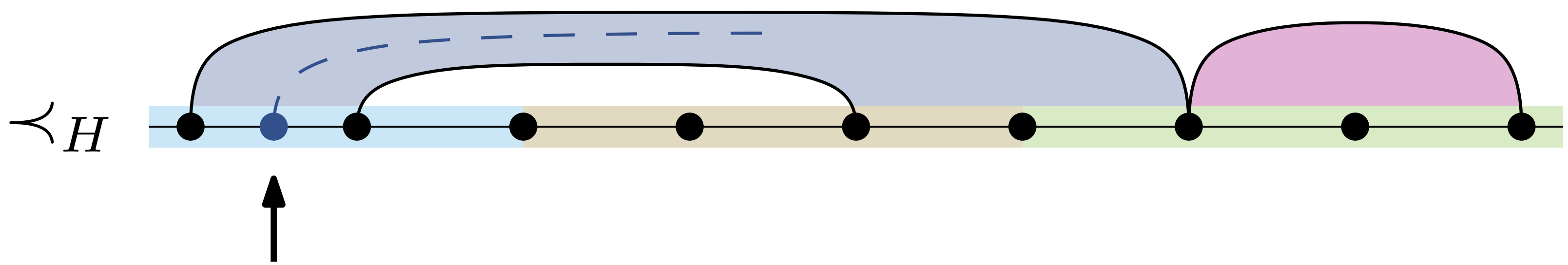
Number faces from outside-in $\Rightarrow$ numbers bounded by ω

Step 4: Guess for each new edge its distance to the outerface

Step 5: Apply a DP in each branch to compute stack layout

$\mathcal{O}(n_{\text{add}} m_{\text{add}} |\mathcal{I}|)$ time per branch

A Dynamic Program To Prevent the Remaining Crossings



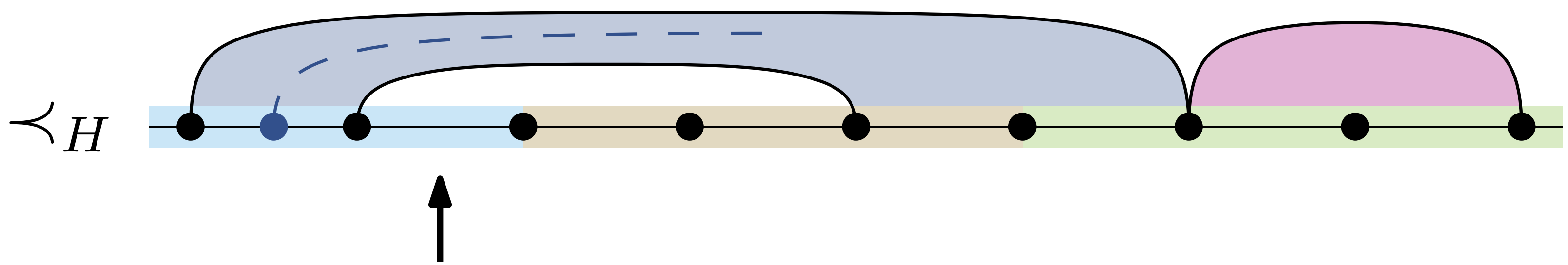
Number faces from outside-in $\Rightarrow$ numbers bounded by ω

Step 4: Guess for each new edge its distance to the outerface

Step 5: Apply a DP in each branch to compute stack layout

$\mathcal{O}(n_{\text{add}} m_{\text{add}} |\mathcal{I}|)$ time per branch

A Dynamic Program To Prevent the Remaining Crossings



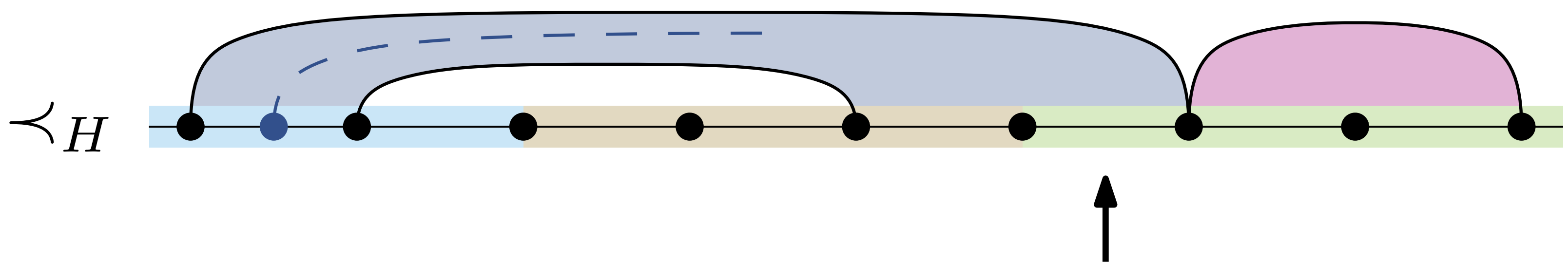
Number faces from outside-in $\Rightarrow$ numbers bounded by ω

Step 4: Guess for each new edge its distance to the outerface

Step 5: Apply a DP in each branch to compute stack layout

$\mathcal{O}(n_{\text{add}} m_{\text{add}} |\mathcal{I}|)$ time per branch

A Dynamic Program To Prevent the Remaining Crossings



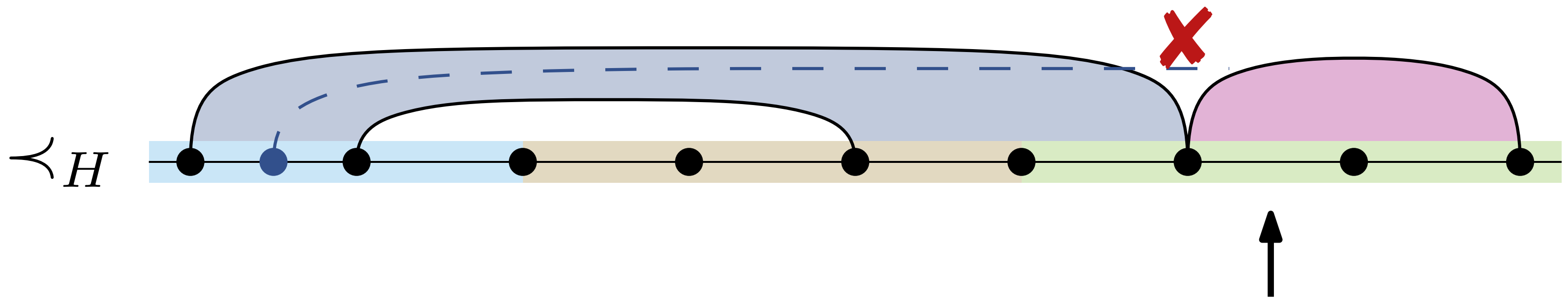
Number faces from outside-in $\Rightarrow$ numbers bounded by ω

Step 4: Guess for each new edge its distance to the outerface

Step 5: Apply a DP in each branch to compute stack layout

$\mathcal{O}(n_{\mathsf{add}} m_{\mathsf{add}} |\mathcal{I}|)$ time per branch

A Dynamic Program To Prevent the Remaining Crossings



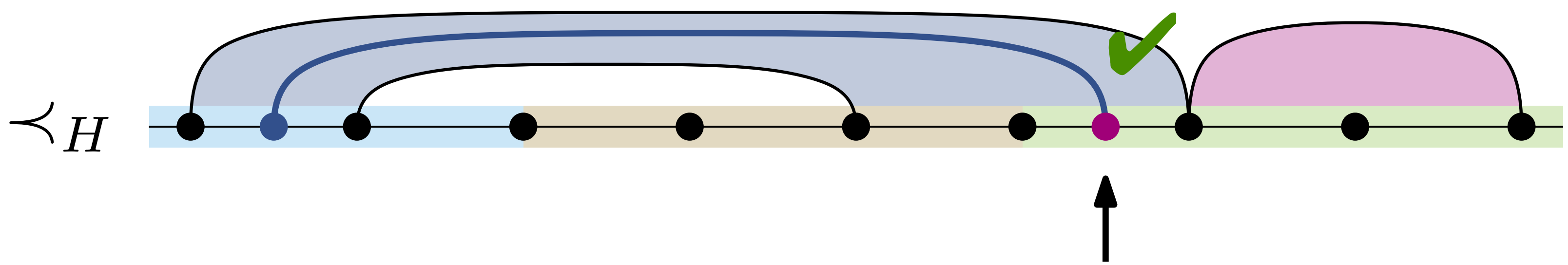
Number faces from outside-in $\Rightarrow$ numbers bounded by ω

Step 4: Guess for each new edge its distance to the outerface

Step 5: Apply a DP in each branch to compute stack layout

$\mathcal{O}(n_{\text{add}} m_{\text{add}} |\mathcal{I}|)$ time per branch

A Dynamic Program To Prevent the Remaining Crossings



Number faces from outside-in $\Rightarrow$ numbers bounded by ω

Step 4: Guess for each new edge its distance to the outerface

Step 5: Apply a DP in each branch to compute stack layout

$\mathcal{O}(n_{\text{add}} m_{\text{add}} |\mathcal{I}|)$ time per branch