



# Precise Complexity Analysis of Coverability in Vector Addition Systems

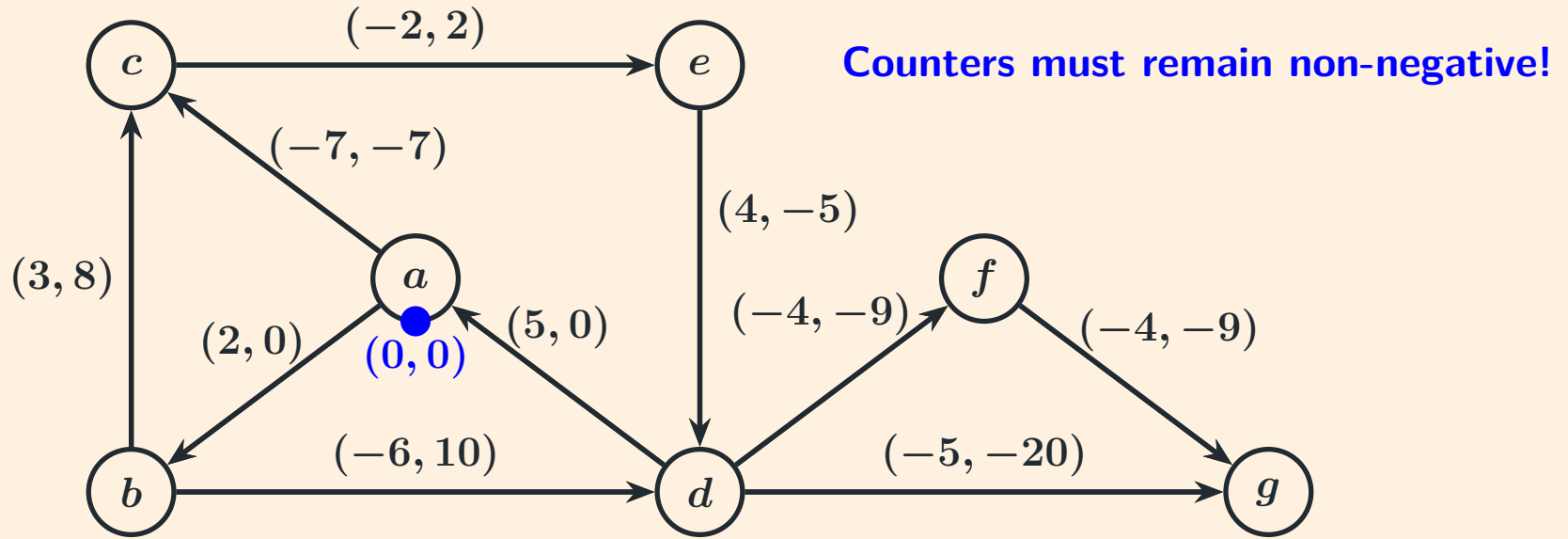
Henry Sinclair-Banks  
MPI-SWS, Kaiserslautern, Germany

INFINITY 2026: Fine-grained Questions in Automata Theory and Verification

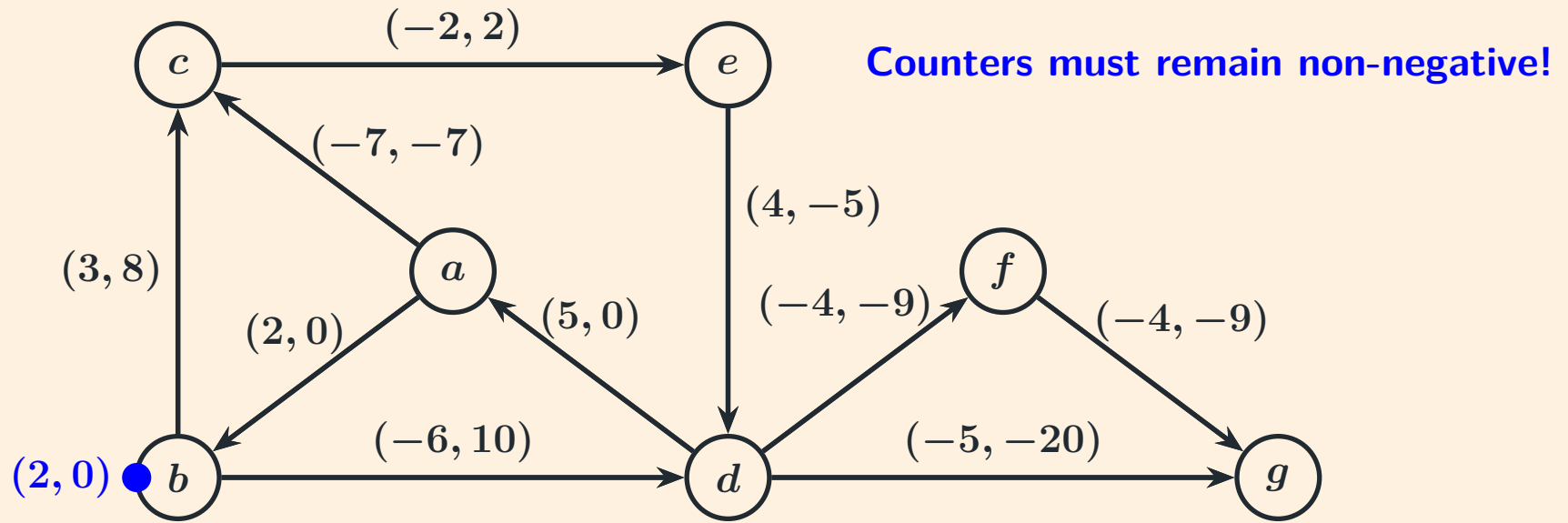
6th July 2026

Windsor 0-02, Royal Holloway, University of London, Egham, UK

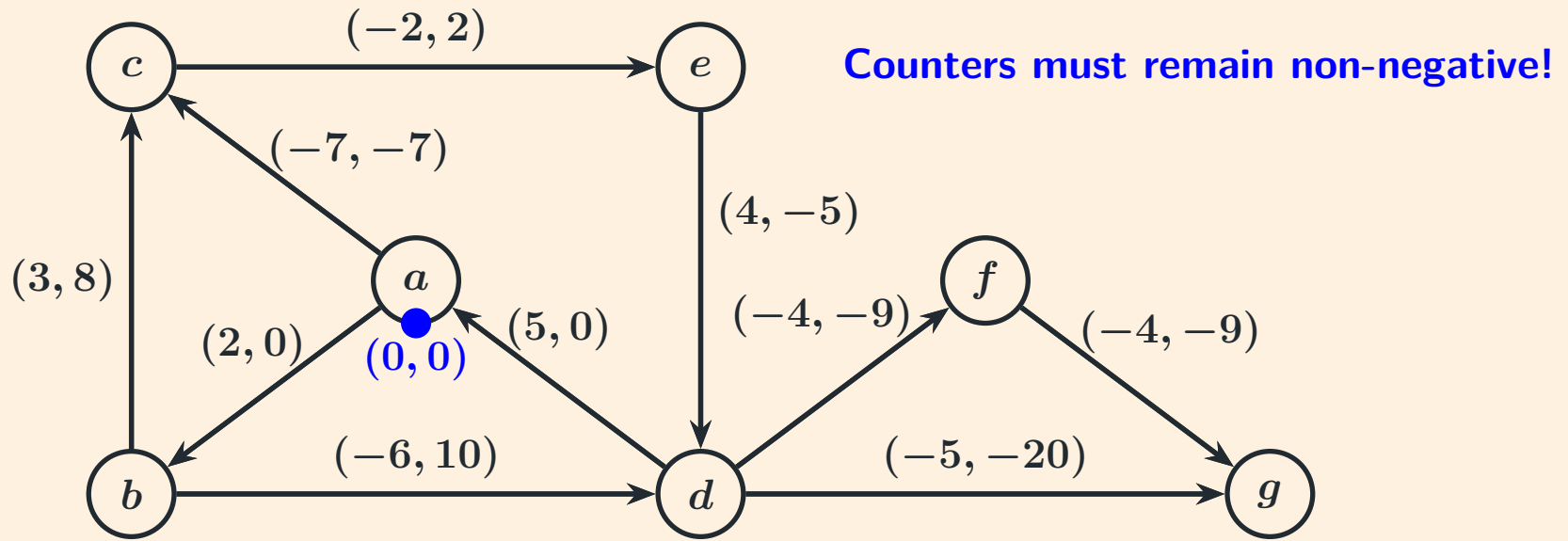
# ~~2-dimensional Vector Addition Systems with States~~ 2-VASS



# ~~2-dimensional Vector Addition Systems with States~~ 2-VASS



# Coverability in 2-VASS



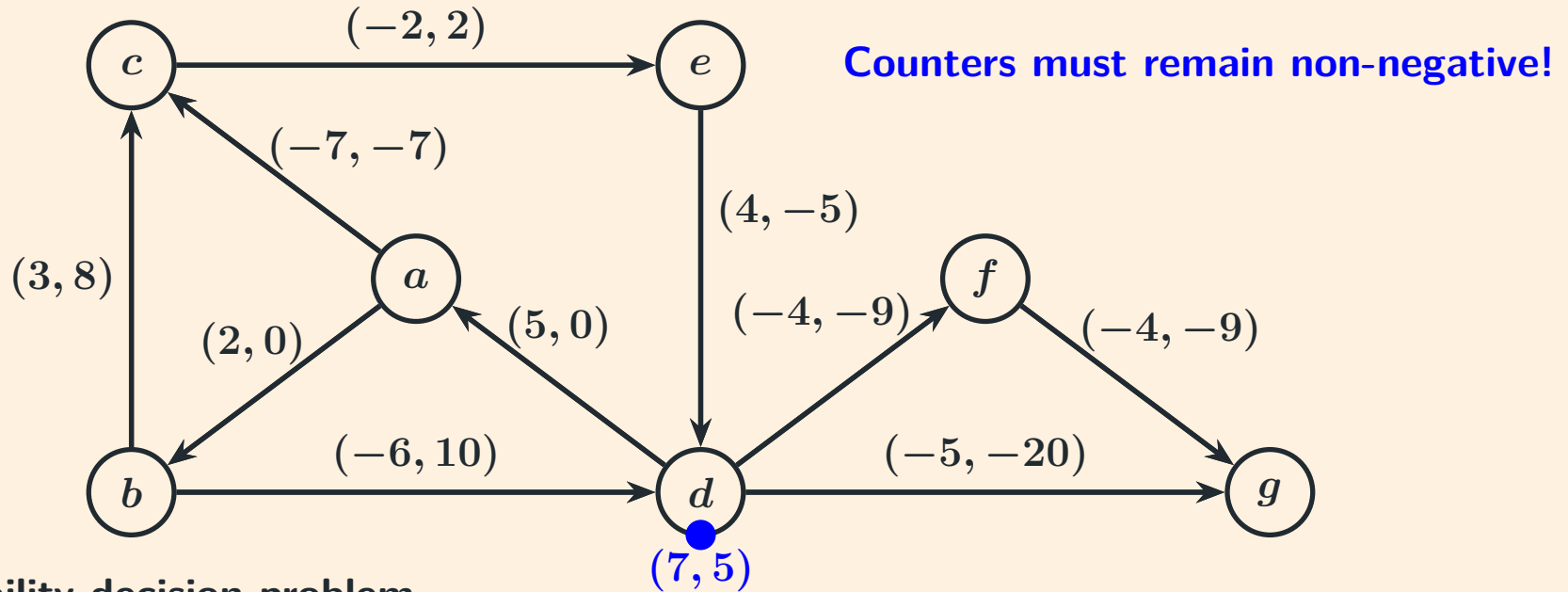
**Coverability decision problem.**

**Input:** A VASS, an initial configuration  $(s, \vec{x})$ , and a target configuration  $(t, \vec{y})$ .

**Question:** Does there exist a run from  $(s, \vec{x})$  to  $(t, \vec{y}')$ , for some  $\vec{y}' \geq \vec{y}$ ?

**Example:** does there exist a run from  $(a, (0, 0))$  that covers  $(g, (0, 0))$ ?

# Coverability in 2-VASS



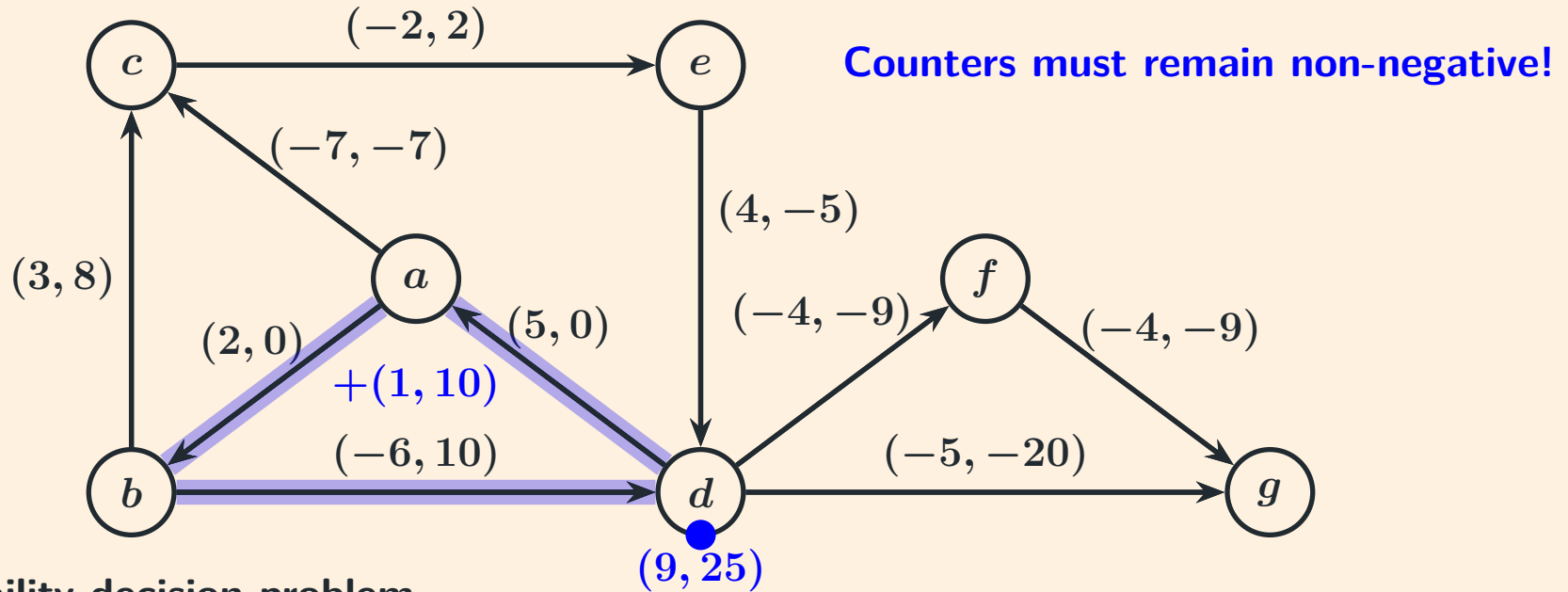
**Coverability decision problem.**

**Input:** A VASS, an initial configuration  $(s, \vec{x})$ , and a target configuration  $(t, \vec{y})$ .

**Question:** Does there exist a run from  $(s, \vec{x})$  to  $(t, \vec{y}')$ , for some  $\vec{y}' \geq \vec{y}$ ?

**Example:** does there exist a run from  $(a, (0, 0))$  that covers  $(g, (0, 0))$ ?

## Coverability in 2-VASS



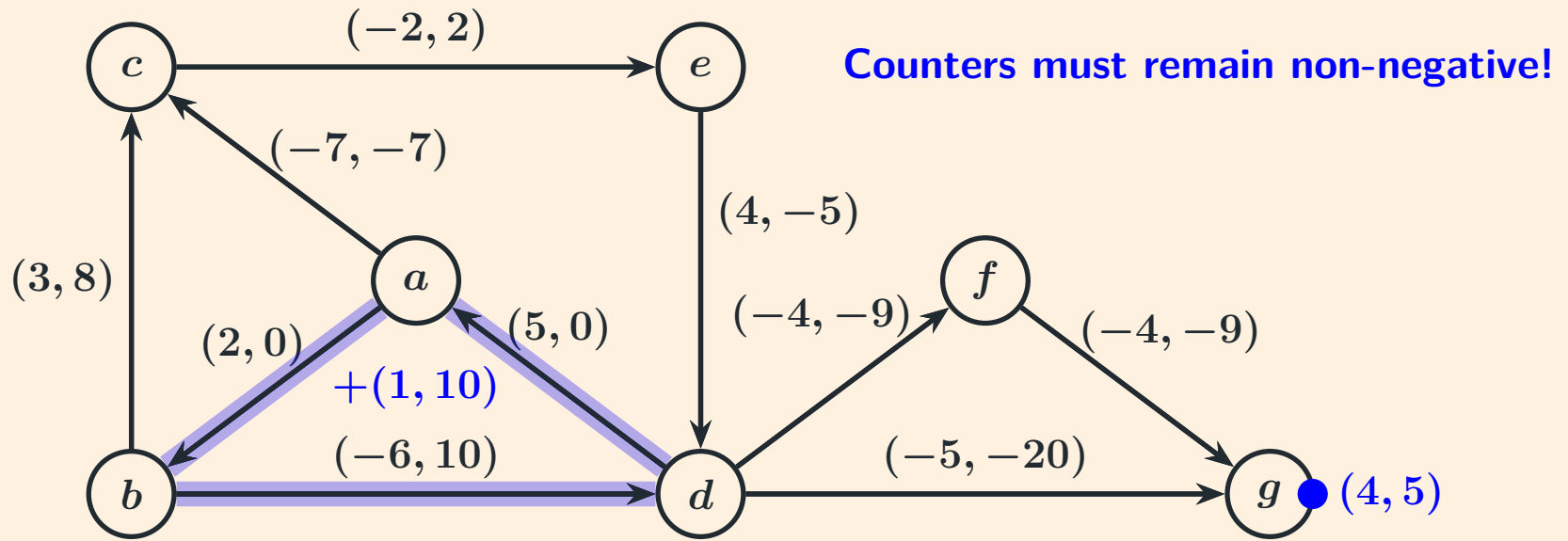
**Coverability decision problem.**

**Input:** A VASS, an initial configuration  $(s, \vec{x})$ , and a target configuration  $(t, \vec{y})$ .

**Question:** Does there exist a run from  $(s, \vec{x})$  to  $(t, \vec{y}')$ , for some  $\vec{y}' \geq \vec{y}$ ?

**Example:** does there exist a run from  $(a, (0, 0))$  that covers  $(g, (0, 0))$ ?

## Coverability in 2-VASS



**Coverability decision problem.**

**Input:** A VASS, an initial configuration  $(s, \vec{x})$ , and a target configuration  $(t, \vec{y})$ .

**Question:** Does there exist a run from  $(s, \vec{x})$  to  $(t, \vec{y}')$ , for some  $\vec{y}' \geq \vec{y}$ ?

**Example:** does there exist a run from  $(a, (0, 0))$  that covers  $(g, (0, 0))$ ?

For the remainder of this talk...

$d$  is the dimension (number of counters).

$n$  is the size of the instance in unary.

# Preview of this talk

## Part 1: $d$ -VASS

Coverability in  $d$ -VASS can be decided in  $n^{2^{\mathcal{O}(d)}}$  time.

Coverability in  $d$ -VASS conditionally requires  $n^{2^{\Omega(d)}}$  time.

Parameterised reduction from  $k$ -clique detection.

## Part 2: 2-VASS

Coverability in 2-VASS can be decided in  $n^C$  time.

Coverability in 2-VASS conditionally requires  $n^2$  time.

Linear-time reduction from  $k$ -cycle detection.

## Part 3: 1-VASS

Coverability in 1-VASS can be decided in  $n^2$  time

Coverability in (binary) 1-VASS is in  $AC^1$ .

### Open Problem

What is the *exact* complexity of coverability in 1-VASS?

# The Complexity of Coverability in $d$ -VASS

**Theorem.** In a  $d$ -VASS, if there is a run from  $(s, \vec{x})$  that covers  $(t, \vec{y})$ , then there is a run from  $(s, \vec{x})$  that covers  $(t, \vec{y})$  of length at most  $n^{2^{O(d)}}$ . [Künnemann, Mazowiecki, Schütze, S-B, Węgrzycki '23]  
[Rackoff '78]

**Corollary.** Coverability in  $d$ -VASS can be decided in  $n^{2^{O(d)}}$  time.

# The Complexity of Coverability in $d$ -VASS

**Theorem.** In a  $d$ -VASS, if there is a run from  $(s, \vec{x})$  that covers  $(t, \vec{y})$ , then there is a run from  $(s, \vec{x})$  that covers  $(t, \vec{y})$  of length at most  $n^{2^{O(d)}}$ .

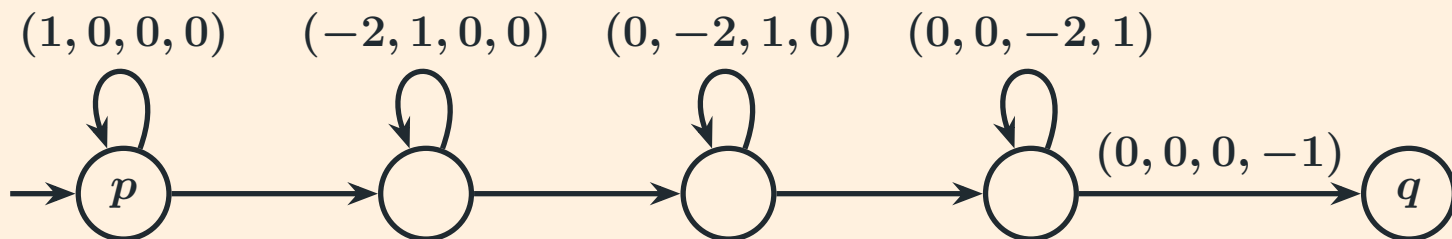
[Künnemann, Mazowiecki, Schütze, S-B, Węgrzycki '23]

[Rackoff '78]

**Corollary.** Coverability in  $d$ -VASS can be decided in  $n^{2^{O(d)}}$  time.

**Theorem.** There exists a family of instance of coverability in  $d$ -VASS for which the shortest covering runs have length  $n^{2^{\Omega(d)}}$ .

[Lipton '76]



# The Complexity of Coverability in $d$ -VASS

**Theorem.** In a  $d$ -VASS, if there is a run from  $(s, \vec{x})$  that covers  $(t, \vec{y})$ , then there is a run from  $(s, \vec{x})$  that covers  $(t, \vec{y})$  of length at most  $n^{2^{O(d)}}$ . [Künnemann, Mazowiecki, Schütze, S-B, Węgrzycki '23]  
[Rackoff '78]

**Corollary.** Coverability in  $d$ -VASS can be decided in  $n^{2^{O(d)}}$  time.

**Theorem.** There exists a family of instance of coverability in  $d$ -VASS for which the shortest covering runs have length  $n^{2^{\Omega(d)}}$ . [Lipton '76]

 does not directly imply :(

**Theorem.** Deciding coverability in  $d$ -VASS requires  $n^{2^{\Omega(d)}}$  time.

# Conditionally Optimal Lower Bound for Coverability

**Theorem.** Assuming the Exponential Time Hypothesis, coverability in  $d$ -VASS requires  $n^{2^{\Omega(d)}}$  time.

[Künnemann, Mazowiecki, Schütze, S-B, and Węgrzycki '23]

**Exponential Time Hypothesis (ETH):** 3-SAT with  $k$ -variables requires  $2^{\Omega(k)}$  time.



Detecting whether there is a  $k$ -clique in an  $n$ -vertex graph requires  $n^{\Omega(k)}$  time.

# Conditionally Optimal Lower Bound for Coverability

**Theorem.** Assuming the Exponential Time Hypothesis, coverability in  $d$ -VASS requires  $n^{2^{\Omega(d)}}$  time.

[Künnemann, Mazowiecki, Schütze, S-B, and Węgrzycki '23]

**Exponential Time Hypothesis (ETH):** 3-SAT with  $k$ -variables requires  $2^{\Omega(k)}$  time.



Detecting whether there is a  $k$ -clique in a  $k$ -partite  $n$ -vertex graph requires  $n^{\Omega(k)}$  time.

[Chen, Chor, Fellows, Huang, Juedes, Kanj, and Xia '05]

[Chen, Huang, Kanj, and Xia '06]

[Cygan, Fomin, Kowalik, Lokshtanov, Marx, Ma. Pilipczuk, and Mi. Pilipczuk '15]

# Conditionally Optimal Lower Bound for Coverability

**Theorem.** Assuming the Exponential Time Hypothesis, coverability in  $d$ -VASS requires  $n^{2^{\Omega(d)}}$  time.

[Künnemann, Mazowiecki, Schütze, S-B, and Węgrzycki '23]

**Idea.** Reduce detecting a  $2^d$ -clique in a  $2^d$ -partite  $n$ -vertex graph to coverability.

**Exponential Time Hypothesis (ETH):** 3-SAT with  $k$ -variables requires  $2^{\Omega(k)}$  time.



Detecting whether there is a  $k$ -clique in a  $k$ -partite  $n$ -vertex graph requires  $n^{\Omega(k)}$  time.

[Chen, Chor, Fellows, Huang, Juedes, Kanj, and Xia '05]

[Chen, Huang, Kanj, and Xia '06]

[Cygan, Fomin, Kowalik, Lokshtanov, Marx, Ma. Pilipczuk, and Mi. Pilipczuk '15]

## Proof Idea: Use Bounded Two-Counter Machines

**Idea:** Reduce detecting a  $2^d$ -clique in a  $2^d$ -partite  $n$ -vertex directed graph to coverability.

First, reduce to coverability in a  $n^{2^{\mathcal{O}(d)}}$ -bounded two-counter machine.

Then, simulate a  $n^{2^{\mathcal{O}(d)}}$ -bounded two-counter machine using an  $\mathcal{O}(n)$ -state  $\mathcal{O}(d)$ -VASS.

# Proof Idea: Use Bounded Two-Counter Machines

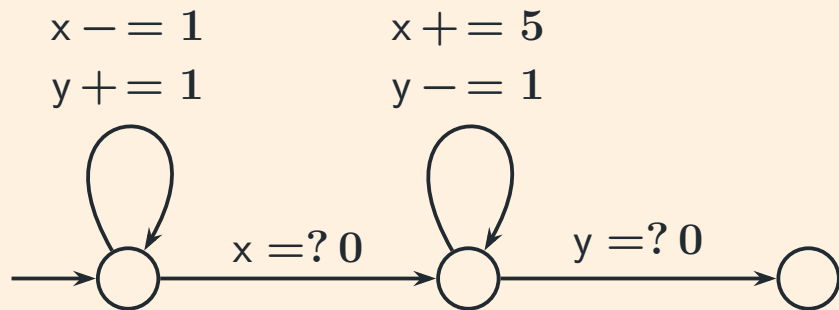
**Idea:** Reduce detecting a  $2^d$ -clique in a  $2^d$ -partite  $n$ -vertex directed graph to coverability.

First, reduce to coverability in a  $n^{2^{\mathcal{O}(d)}}$ -bounded two-counter machine.

Then, simulate a  $n^{2^{\mathcal{O}(d)}}$ -bounded two-counter machine using an  $\mathcal{O}(n)$ -state  $\mathcal{O}(d)$ -VASS.

An  $n^{2^{\mathcal{O}(d)}}$ -bounded two-counter machine has two counters  $x, y \in \{0, 1, \dots, n^{2^{\mathcal{O}(d)}}\}$  that can be added to ( $x += 2$ ), subtracted from ( $y -= 3$ ), and zero-tested ( $x =? 0$ ).

1. LOOP( $x -= 1, y += 1$ )
2.  $x =? 0$
3. LOOP( $x += 5, y -= 1$ )
4.  $y =? 0$



# Proof Idea: Use Bounded Two-Counter Machines

**Idea:** Reduce detecting a  $2^d$ -clique in a  $2^d$ -partite  $n$ -vertex directed graph to coverability.

First, reduce to coverability in a  $n^{2^{\mathcal{O}(d)}}$ -bounded two-counter machine.

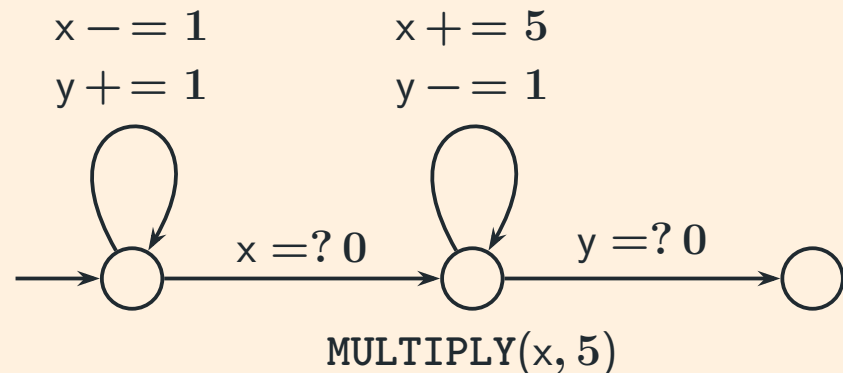
Then, simulate a  $n^{2^{\mathcal{O}(d)}}$ -bounded two-counter machine using an  $\mathcal{O}(n)$ -state  $\mathcal{O}(d)$ -VASS.

An  $n^{2^{\mathcal{O}(d)}}$ -bounded two-counter machine has two counters  $x, y \in \{0, 1, \dots, n^{2^{\mathcal{O}(d)}}\}$  that can be added to ( $x += 2$ ), subtracted from ( $y -= 3$ ), and zero-tested ( $x =? 0$ ).

Pre:  $x = x, y = 0$

1. LOOP( $x -= 1, y += 1$ )
2.  $x =? 0$
3. LOOP( $x += 5, y -= 1$ )
4.  $y =? 0$

Post:  $x = x \cdot 5, y = 0$



# Proof Idea: Use Bounded Two-Counter Machines

**Idea:** Reduce detecting a  $2^d$ -clique in a  $2^d$ -partite  $n$ -vertex directed graph to coverability.

First, reduce to coverability in a  $n^{2^{\mathcal{O}(d)}}$ -bounded two-counter machine.

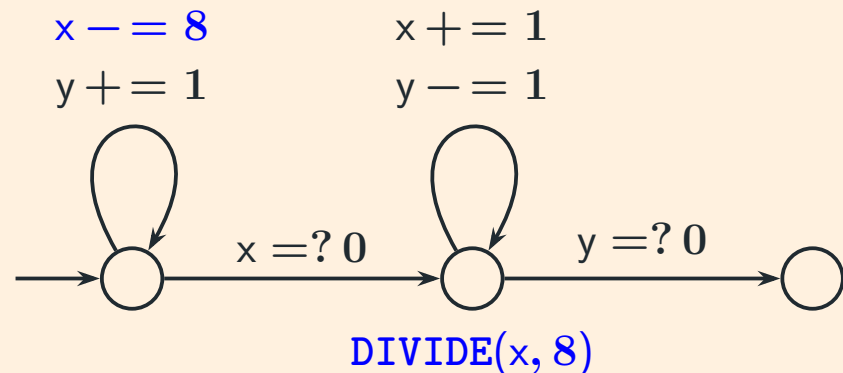
Then, simulate a  $n^{2^{\mathcal{O}(d)}}$ -bounded two-counter machine using an  $\mathcal{O}(n)$ -state  $\mathcal{O}(d)$ -VASS.

An  $n^{2^{\mathcal{O}(d)}}$ -bounded two-counter machine has two counters  $x, y \in \{0, 1, \dots, n^{2^{\mathcal{O}(d)}}\}$  that can be added to ( $x += 2$ ), subtracted from ( $y -= 3$ ), and zero-tested ( $x =? 0$ ).

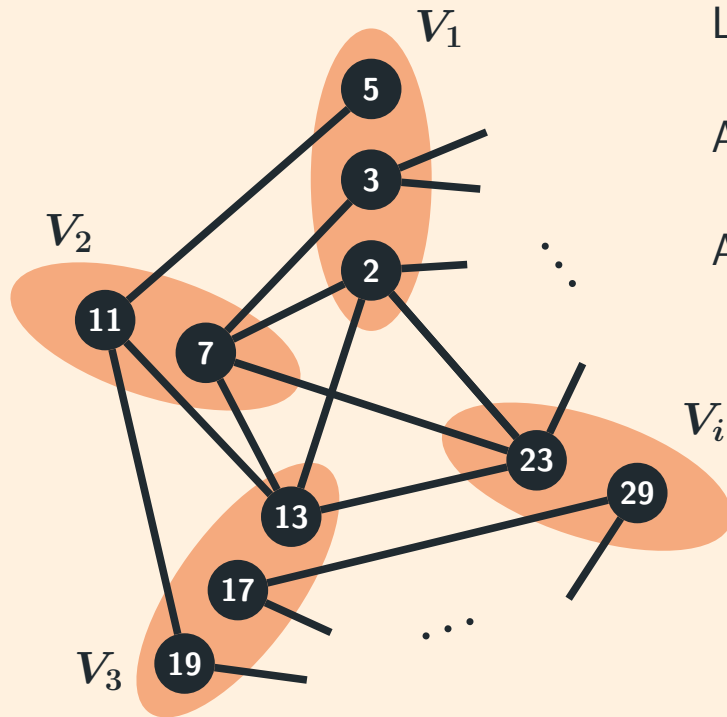
Pre:  $x = x, y = 0$

1. LOOP( $x -= 8, y += 1$ )
2.  $x =? 0$
3. LOOP( $x += 1, y -= 1$ )
4.  $y =? 0$

Post:  $x = x \div 8, y = 0$



# Detecting Cliques using Divisibility Tests

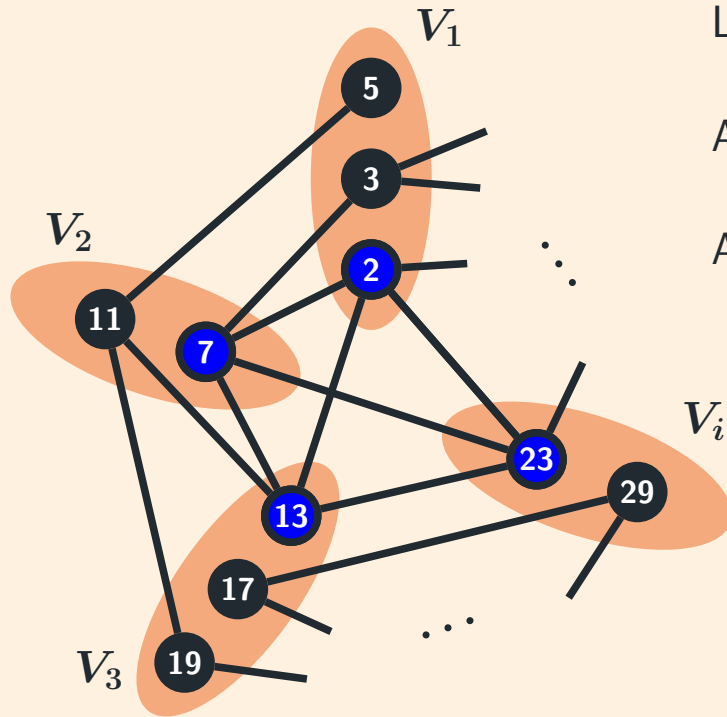


Let  $(V_1 \cup V_2 \cup \dots \cup V_k, E)$  be a  $k$ -partite  $n$ -vertex graph.

Associate the first  $n$  primes with the vertices.

A candidate  $k$ -clique is represented by a product of  $k$  primes.

# Detecting Cliques using Divisibility Tests



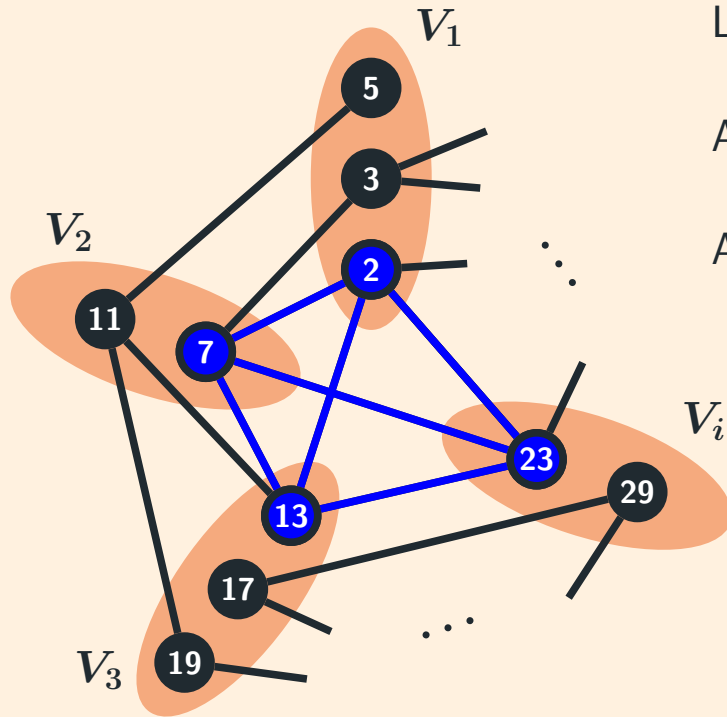
Let  $(V_1 \cup V_2 \cup \dots \cup V_k, E)$  be a  $k$ -partite  $n$ -vertex graph.

Associate the first  $n$  primes with the vertices.

A candidate  $k$ -clique is represented by a product of  $k$  primes.

Example:  $c = 2 \cdot 7 \cdot 13 \cdot \dots \cdot 23$ .

# Detecting Cliques using Divisibility Tests



Let  $(V_1 \cup V_2 \cup \dots \cup V_k, E)$  be a  $k$ -partite  $n$ -vertex graph.

Associate the first  $n$  primes with the vertices.

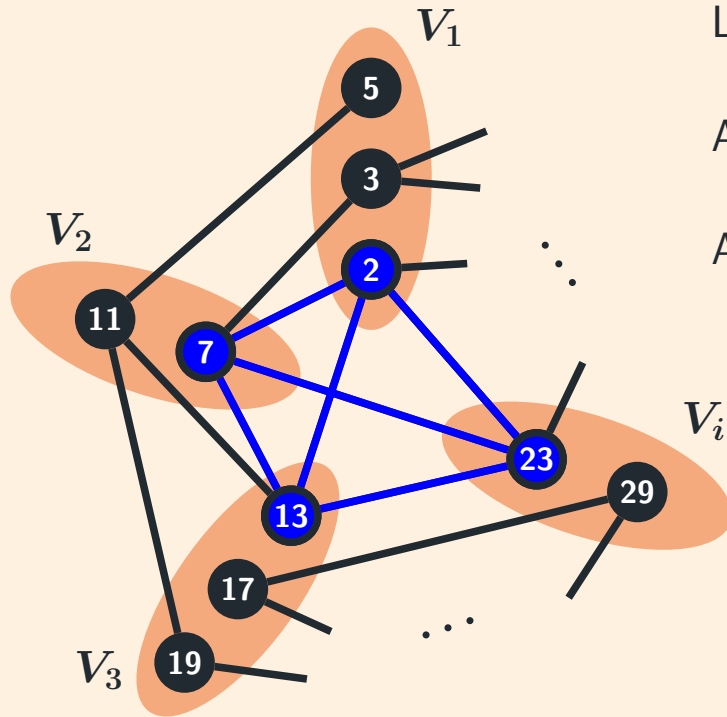
A candidate  $k$ -clique is represented by a product of  $k$  primes.

Example:  $c = 2 \cdot 7 \cdot 13 \cdot \dots \cdot 23$ .

To check if  $v$  represents a clique, use divisibility tests to verify all nodes are adjacent.

Example:  $(2 \cdot 7) | c?$   $(2 \cdot 13) | c?$   $(7 \cdot 13) | c?$  ...  
 $(2 \cdot 23) | c?$   $(7 \cdot 23) | c?$   $(13 \cdot 23) | c?$

# Detecting Cliques using Divisibility Tests



Let  $(V_1 \cup V_2 \cup \dots \cup V_k, E)$  be a  $k$ -partite  $n$ -vertex graph.

Associate the first  $n$  primes with the vertices.

A candidate  $k$ -clique is represented by a product of  $k$  primes.

Example:  $c = 2 \cdot 7 \cdot 13 \cdot \dots \cdot 23$ .

To check if  $v$  represents a clique, use divisibility tests to verify all nodes are adjacent.

Example:  $(2 \cdot 7) | c?$   $(2 \cdot 13) | c?$   $(7 \cdot 13) | c?$  ...  
 $(2 \cdot 23) | c?$   $(7 \cdot 23) | c?$   $(13 \cdot 23) | c?$

There exist  $p_1 \in \text{Primes}(V_1), \dots, p_k \in \text{Primes}(V_k)$  such that, for every pair  $1 \leq i < j \leq k$ , there is an edge  $\{p, q\} \in (V_i \times V_j) \cap E$  such that  $(p \cdot q) | p_1 \cdot \dots \cdot p_k \iff$  there exists a  $k$ -clique.

# Bounded Two-Counter Machine Implementation

There exist  $p_1 \in \text{Primes}(V_1), \dots, p_k \in \text{Primes}(V_k)$  such that for every pair  $1 \leq i < j \leq k$ , there is an edge  $\{p, q\} \in (V_i \times V_j) \cap E$  such that  $(p \cdot q) \mid p_1 \cdot \dots \cdot p_k \iff$  there exists a  $k$ -clique.

# Bounded Two-Counter Machine Implementation

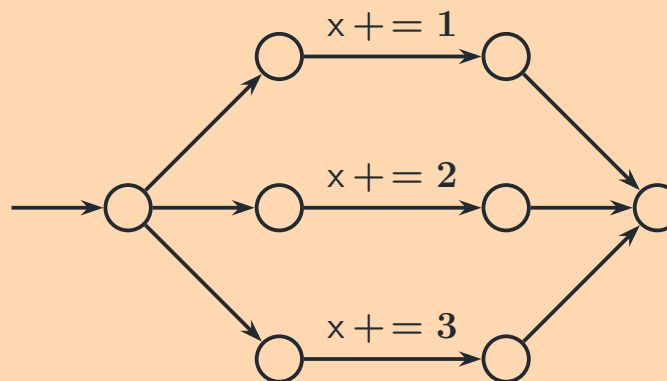
There  $\epsilon$   
is an ec

## Guessing with Nondeterministic Branching

Pre:  $x = x$

1. GUESS  $c \in \{1, 2, 3\}$
2.  $x += c$

Post:  $x = x + 1$ , or  
 $x = x + 2$ , or  
 $x = x + 3$ .



$k$ , there  
 $k$ -clique.

# Bounded Two-Counter Machine Implementation

There exist  $p_1 \in \text{Primes}(V_1), \dots, p_k \in \text{Primes}(V_k)$  such that for every pair  $1 \leq i < j \leq k$ , there is an edge  $\{p, q\} \in (V_i \times V_j) \cap E$  such that  $(p \cdot q) \mid p_1 \cdot \dots \cdot p_k \iff$  there exists a  $k$ -clique.

Part one: Guess a candidate clique.

Pre:  $x = 1, y = 0$ .

↓  
GUESS  $p_1 \in \text{Primes}(V_1)$   
MULTIPLY( $x, p_1$ )  
⋮  
GUESS  $p_k \in \text{Primes}(V_k)$   
MULTIPLY( $x, p_k$ )  
↓

Post:  $x = p_1 \cdot \dots \cdot p_k, y = 0$ .

# Bounded Two-Counter Machine Implementation

There exist  $p_1 \in \text{Primes}(V_1), \dots, p_k \in \text{Primes}(V_k)$  such that for every pair  $1 \leq i < j \leq k$ , there is an edge  $\{p, q\} \in (V_i \times V_j) \cap E$  such that  $(p \cdot q) \mid p_1 \cdot \dots \cdot p_k \iff$  there exists a  $k$ -clique.

Part one: Guess a candidate clique.

Pre:  $x = 1, y = 0$ .

GUESS  $p_1 \in \text{Primes}(V_1)$

MULTIPLY( $x, p_1$ )

$\vdots$

GUESS  $p_k \in \text{Primes}(V_k)$

MULTIPLY( $x, p_k$ )

Post:  $x = p_1 \cdot \dots \cdot p_k, y = 0$ .

Part two: Check the candidate is a clique.

Pre:  $x = p_1 \cdot \dots \cdot p_k, y = 0$ .

GUESS  $\{p_1, p_2\} \in (V_1 \times V_2) \cap E$

DIVIDE( $x, p_1 \cdot p_2$ )

MULTIPLY( $x, p_1 \cdot p_2$ )

$\vdots$

GUESS  $\{p_{k-1}, p_k\} \in (V_{k-1} \times V_k) \cap E$

DIVIDE( $x, p_{k-1} \cdot p_k$ )

MULTIPLY( $x, p_{k-1} \cdot p_k$ )

Post:  $x = p_1 \cdot \dots \cdot p_k, y = 0$ .

# Bounded Two-Counter Machine Implementation

There exist  $p_1 \in \text{Primes}(V_1), \dots, p_k \in \text{Primes}(V_k)$  such that for every pair  $1 \leq i < j \leq k$ , there is an edge  $\{p, q\} \in (V_i \times V_j) \cap E$  such that  $(p \cdot q) \mid p_1 \cdot \dots \cdot p_k \iff$  there exists a  $k$ -clique.

Part one: Guess a candidate clique.

Pre:  $x = 1, y = 0$ .

GUESS  $p_1 \in \text{Primes}(V_1)$

MULTIPLY( $x, p_1$ )

GUESS  $p_k \in \text{Primes}(V_k)$

MULTIPLY( $x, p_k$ )

Post:  $x = p_1 \cdot \dots \cdot p_k, y = 0$ .

Part two: Check the candidate is a clique.

Pre:  $x = p_1 \cdot \dots \cdot p_k, y = 0$

DIVIDE( $x, p_1 \cdot p_2$ )

MULTIPLY( $x, p_1 \cdot p_2$ )

$\vdots$

GUESS  $\{p_{k-1}, p_k\} \in (V_{k-1} \times V_k) \cap E$

DIVIDE( $x, p_{k-1} \cdot p_k$ )

MULTIPLY( $x, p_{k-1} \cdot p_k$ )

Post:  $x = p_1 \cdot \dots \cdot p_k, y = 0$ .

This two-counter program terminates  $\iff$  there exists a  $k$ -clique.

## VASS can Simulate Bounded Two-Counter Machines

Counter bound of  $k$ -clique detecting two-counter machine:  $p_{\max}^k \leq \mathcal{O}(n^k \log(n)^k) \leq \mathcal{O}(n^{2k})$ .

Size of  $k$ -clique detecting two-counter machine:  $\mathcal{O}(n^{11}) \leq \text{poly}(n)$ .

# VASS can Simulate Bounded Two-Counter Machines

Counter bound of  $k$ -clique detecting two-counter machine:  $p_{\max}^k \leq \mathcal{O}(n^k \log(n)^k) \leq \mathcal{O}(n^{2k})$ .

Size of  $k$ -clique detecting two-counter machine:  $\mathcal{O}(n^{11}) \leq \text{poly}(n)$ .

**Lemma.** In  $\text{poly}(n)$  time, one can construct a  $\mathcal{O}(\log(k))$ -VASS that can *simulate* an  $\mathcal{O}(n^k)$ -bounded two-counter machine of  $\text{poly}(n)$  size. [Rosier and Yen '85]

# VASS can Simulate Bounded Two-Counter Machines

Counter bound of  $k$ -clique detecting two-counter machine:  $p_{\max}^k \leq \mathcal{O}(n^k \log(n)^k) \leq \mathcal{O}(n^{2k})$ .

Size of  $k$ -clique detecting two-counter machine:  $\mathcal{O}(n^{11}) \leq \text{poly}(n)$ .

**Lemma.** In  $\text{poly}(n)$  time, one can construct a  $\mathcal{O}(\log(k))$ -VASS that can *simulate* an  $\mathcal{O}(n^k)$ -bounded two-counter machine of  $\text{poly}(n)$  size. [Rosier and Yen '85]

If we set  $k = 2^d$ , the  $\text{poly}(n)$ -size two-counter machine for detecting  $2^d$ -cliques is  $\mathcal{O}(n^{2^d})$ -bounded.

$\implies$  In  $\text{poly}(n)$  time, one can construct an  $\mathcal{O}(d)$ -VASS for detecting  $2^d$ -cliques.

# VASS can Simulate Bounded Two-Counter Machines

Counter bound of  $k$ -clique detecting two-counter machine:  $p_{\max}^k \leq \mathcal{O}(n^k \log(n)^k) \leq \mathcal{O}(n^{2k})$ .

Size of  $k$ -clique detecting two-counter machine:  $\mathcal{O}(n^{11}) \leq \text{poly}(n)$ .

**Lemma.** In  $\text{poly}(n)$  time, one can construct a  $\mathcal{O}(\log(k))$ -VASS that can *simulate* an  $\mathcal{O}(n^k)$ -bounded two-counter machine of  $\text{poly}(n)$  size. [Rosier and Yen '85]

If we set  $k = 2^d$ , the  $\text{poly}(n)$ -size two-counter machine for detecting  $2^d$ -cliques is  $\mathcal{O}(n^{2^d})$ -bounded.

$\implies$  In  $\text{poly}(n)$  time, one can construct an  $\mathcal{O}(d)$ -VASS for detecting  $2^d$ -cliques.

**Remark.** Termination is coverability.

This two-counter program terminates  $\iff$  there exists a  $k$ -clique.

*“Can I get to the end of the program with any (at least zero) value on each of the counters?”*

# Reducing Clique Detection to Coverability in VASS

Detecting  $2^d$ -cliques in an  $n$ -vertex graph requires  $n^{\Omega(2^d)}$  time under the Exponential Time Hypothesis.



Via divisibility tests of the product-of-primes encoding.

Construct an instance of termination in a  $\text{poly}(n)$ -size  $\mathcal{O}(n^{2^d})$ -bounded two-counter machine.



Using Rosier and Yen's simulation lemma.

Then, in  $\text{poly}(n)$  time, construct an instance of coverability in an  $\mathcal{O}(d)$ -VASS.

**Theorem.** Assuming the exponential time hypothesis, coverability in VASS requires  $n^{2^{\Omega(d)}}$  time.

[Künnemann, Mazowiecki, Schütze, S-B, and Węgrzycki '23]

# Preview of this talk

## Part 1: $d$ -VASS

Coverability in  $d$ -VASS can be decided in  $n^{2^{\mathcal{O}(d)}}$  time.

Coverability in  $d$ -VASS conditionally requires  $n^{2^{\Omega(d)}}$  time.

Parameterised reduction from  $k$ -clique detection.

## Part 2: 2-VASS

Coverability in 2-VASS can be decided in  $n^C$  time.

Coverability in 2-VASS conditionally requires  $n^2$  time.

Linear-time reduction from  $k$ -cycle detection.

## Part 3: 1-VASS

Coverability in 1-VASS can be decided in  $n^2$  time

Coverability in (binary) 1-VASS is in  $AC^1$ .

### Open Problem

What is the *exact* complexity of coverability in 1-VASS?

# Preview of this talk

## Part 1: $d$ -VASS

Coverability in  $d$ -VASS can be decided in  $n^{2^{\mathcal{O}(d)}}$  time.

Coverability in  $d$ -VASS conditionally requires  $n^{2^{\Omega(d)}}$  time.

Parameterised reduction from  $k$ -clique detection.

## Part 2: 2-VASS

Coverability in 2-VASS can be decided in  $n^C$  time.

Coverability in 2-VASS conditionally requires  $n^2$  time.

Linear-time reduction from  $k$ -cycle detection.

## Part 3: 1-VASS

Coverability in 1-VASS can be decided in  $n^2$  time

Coverability in (binary) 1-VASS is in  $AC^1$ .

### Open Problem

What is the *exact* complexity of coverability in 1-VASS?

# The Complexity of Coverability in 2-VASS

**Corollary.** Coverability in  $d$ -VASS can be decided in  $n^{2^{O(d)}}$  time.

[Künnemann, Mazowiecki, Schütze, S-B, and Węgrzycki '23]

**Corollary.** Coverability in 2-VASS can be decided in  $n^{\frac{C}{320}}$  time.

# The Complexity of Coverability in 2-VASS

**Corollary.** Coverability in  $d$ -VASS can be decided in  $n^{2^{\mathcal{O}(d)}}$  time.

[Künnemann, Mazowiecki, Schütze, S-B, and Węgrzycki '23]

**Corollary.** Coverability in 2-VASS can be decided in  $n^{\overset{C}{320}}$  time.

**$k$ -cycle detection problem:** **Input:** A directed graph  $G$ .  
**Question:** Does there exist a cycle in  $G$  of length  $k$ ?

**$k$ -cycle hypothesis:** for every  $\varepsilon > 0$  and  $m$ , there exists  $k$  such that there is no  $\mathcal{O}(m^{2-\varepsilon})$ -time algorithm for detecting  $k$ -cycles in  $m$ -edge graphs.

[Alon, Yuster, and Zwick '97]

[Yuster and Zwick '04]

[Darlirrooyford, Vuong, and Williams '21]

# The Complexity of Coverability in 2-VASS

**Corollary.** Coverability in  $d$ -VASS can be decided in  $n^{2^{\mathcal{O}(d)}}$  time.

[Künnemann, Mazowiecki, Schütze, S-B, and Węgrzycki '23]

**Corollary.** Coverability in 2-VASS can be decided in  $n^{\frac{C}{320}}$  time.

**$k$ -cycle detection problem:** **Input:** A directed graph  $G$ .

**Question:** Does there exist a cycle in  $G$  of length  $k$ ?

**$k$ -cycle hypothesis:** for every  $\varepsilon > 0$  and  $m$ , there exists  $k$  such that there is no  $\mathcal{O}(m^{2-\varepsilon})$ -time algorithm for detecting  $k$ -cycles in  $m$ -edge graphs.

[Alon, Yuster, and Zwick '97]

[Yuster and Zwick '04]

[Darlirrooyford, Vuong, and Williams '21]

**Theorem.** Assuming the  $k$ -cycle hypothesis, there does not exist an  $\varepsilon > 0$  and an  $\mathcal{O}(n^{2-\varepsilon})$ -time algorithm that decides coverability in 2-VASS.

[Künnemann, Mazowiecki, Schütze, S-B, and Węgrzycki '23]

# $k$ -Cycle Hypothesis Simplifying Assumption

**$k$ -cycle hypothesis:** for every  $\varepsilon > 0$  and  $m$ , there exists  $k$  such that there is no  $\mathcal{O}(m^{2-\varepsilon})$ -time algorithm for detecting  $k$ -cycles in  $m$ -edge graphs.

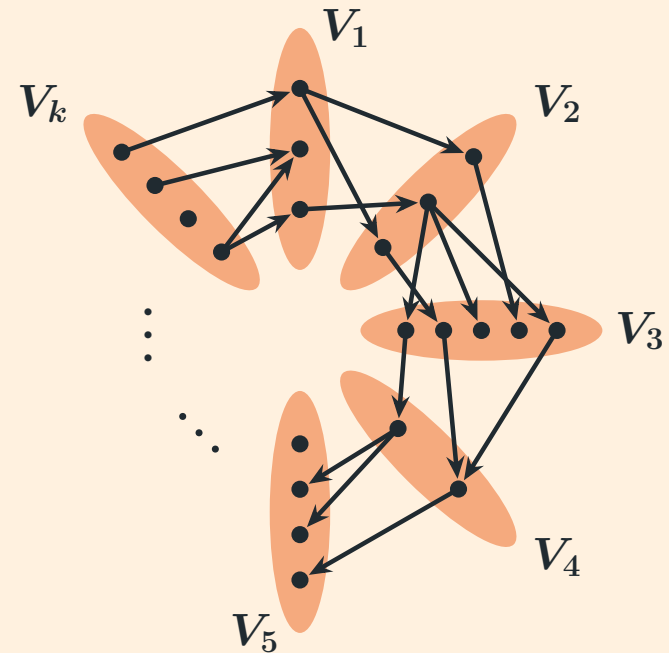
[Alon, Yuster, and Zwick '97]

[Yuster and Zwick '04]

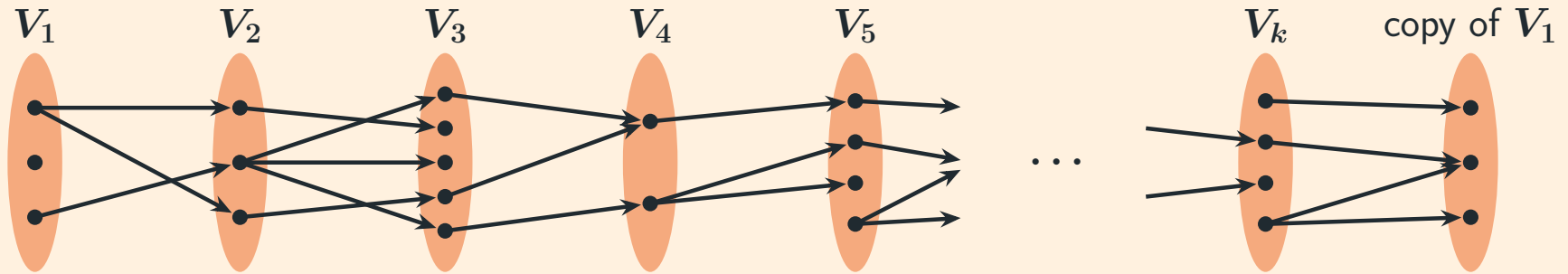
[Darlirrooyford, Vuong, and Williams '21]

It suffices to only consider  **$k$ -circle layered graphs:**

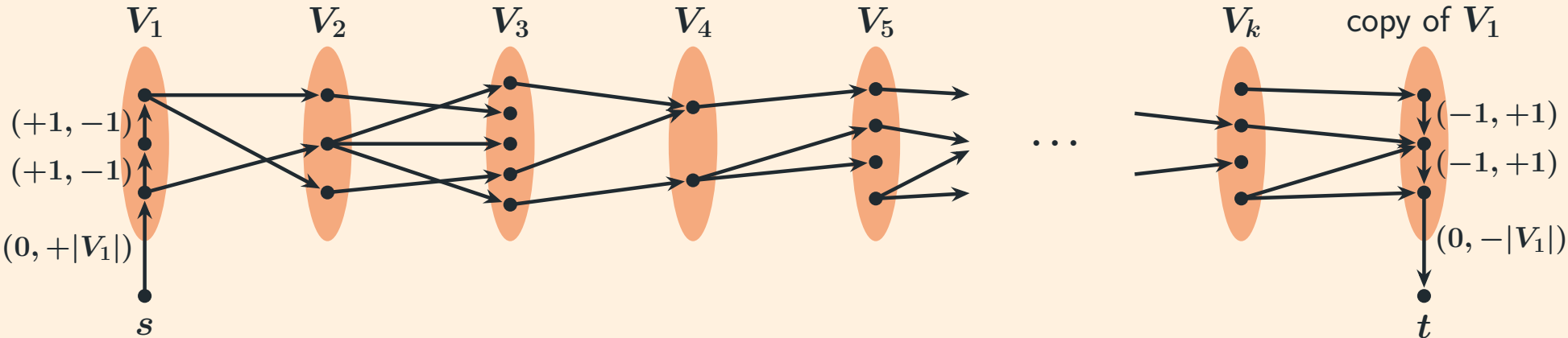
[Lincoln, Williams, and Williams '18]



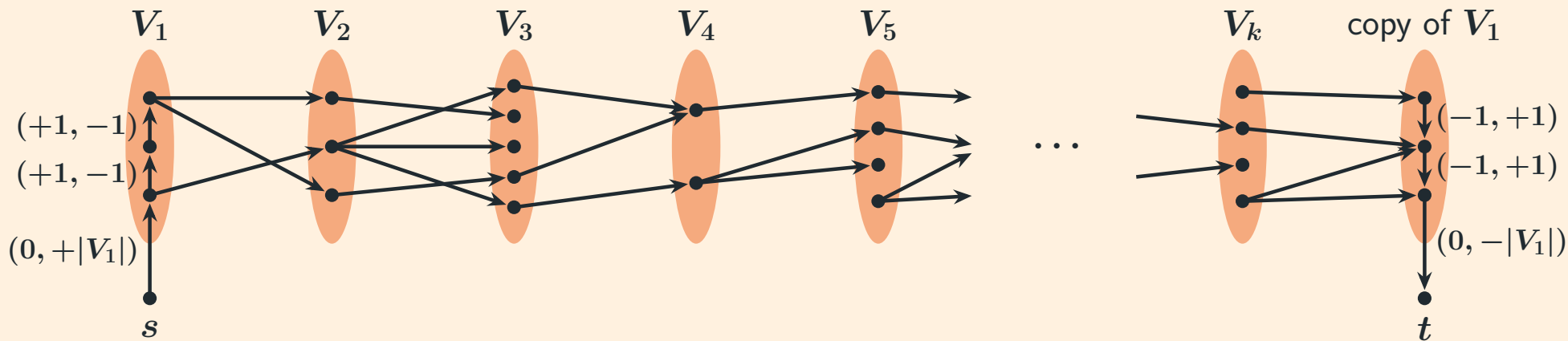
# Reduction From $k$ -Cycle Detection



# Reduction From $k$ -Cycle Detection

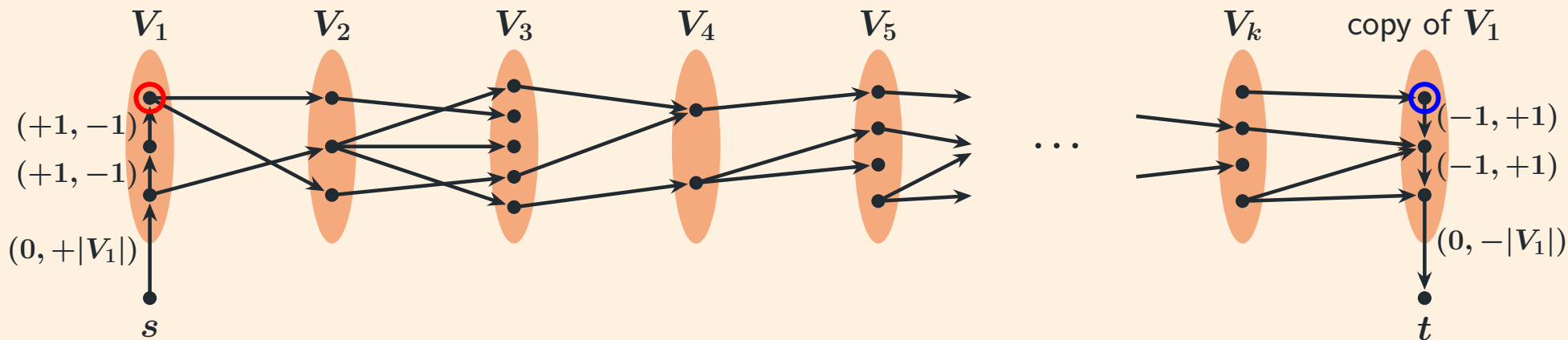


## Reduction From $k$ -Cycle Detection



**Claim.** From  $(s, (0, 0))$  you can cover  $(t, (0, 0)) \iff$  there is a  $k$ -cycle the original graph.

## Reduction From $k$ -Cycle Detection



**Claim.** From  $(s, (0, 0))$  you can cover  $(t, (0, 0)) \iff$  there is a  $k$ -cycle the original graph.

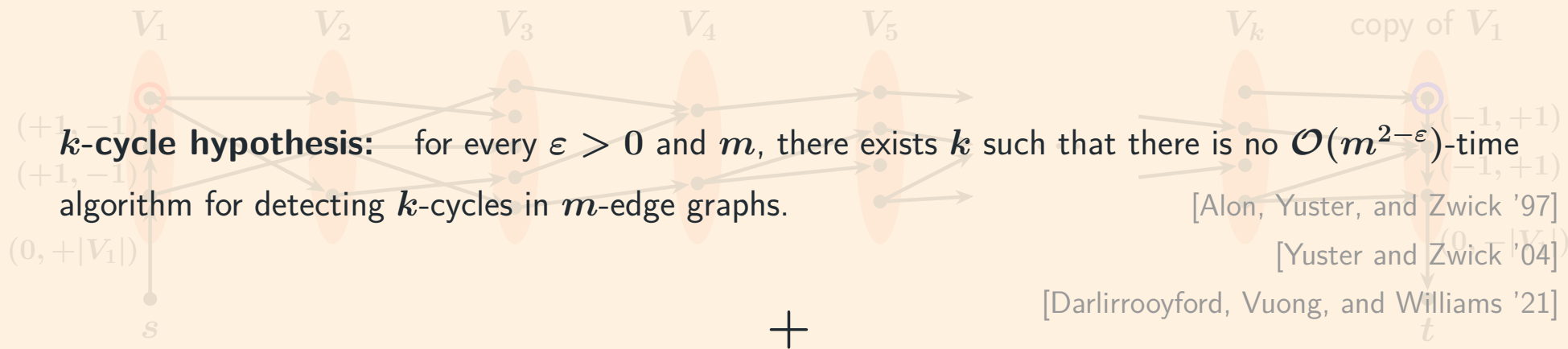
**Idea.** Suppose you leave  $V_1$  via the  $i$ -th node and arrive at the copy of  $V_1$  at the  $j$ -th node.

First component  $\implies i \geq j$ .

Second component  $\implies j \geq i$ .

$\implies$  Any run from  $s$  that reaches  $t$  must have  $i = j$ .

# Reduction From $k$ -Cycle Detection



**Claim.** From  $(s, (0, 0))$  you can cover  $(t, (0, 0)) \iff$  there is a  $k$ -cycle the original graph.

Linear-time reduction!

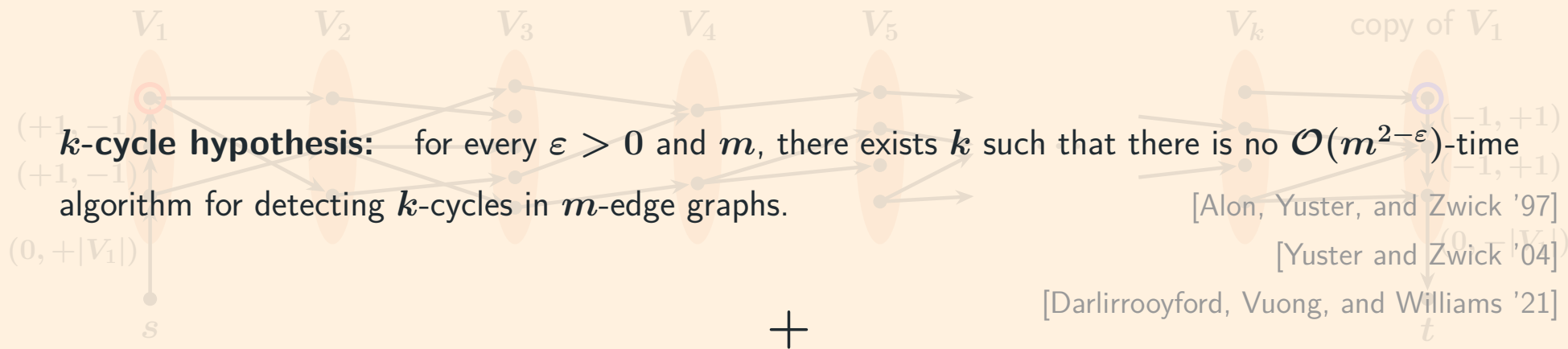
**Idea.** Suppose you leave  $V_1$  via the  $i$ -th node and arrive at the copy of  $V_1$  at the  $j$ -th node.

First component  $\implies i \geq j$ .

Second component  $\implies j \geq i$ .

$\implies$  Any run from  $s$  that reaches  $t$  must have  $i = j$ .

# Reduction From $k$ -Cycle Detection



**Claim.** From  $(s, (0, 0))$  you can cover  $(t, (0, 0)) \iff$  there is a  $k$ -cycle the original graph.

Linear-time reduction!

**Idea.** Suppose you leave  $V_1$  via the  $i$ -th node and arrive at the copy of  $V_1$  at the  $j$ -th node.

First component  $\implies i \geq j$ .

$\implies$  Any run from  $s$  that reaches  $t$  must have  $i = j$ .

**Theorem.** Assuming the  $k$ -cycle hypothesis, there does not exist an  $\epsilon > 0$  and an  $\mathcal{O}(n^{2-\epsilon})$ -time

algorithm that decides coverability in 2-VASS.

[Künnemann, Mazowiecki, Schütze, S-B, and Węgrzycki '23]

# Preview of this talk

## Part 1: $d$ -VASS

Coverability in  $d$ -VASS can be decided in  $n^{2^{\mathcal{O}(d)}}$  time.

Coverability in  $d$ -VASS conditionally requires  $n^{2^{\Omega(d)}}$  time.

Parameterised reduction from  $k$ -clique detection.

## Part 2: 2-VASS

Coverability in 2-VASS can be decided in  $n^C$  time.

Coverability in 2-VASS conditionally requires  $n^2$  time.

Linear-time reduction from  $k$ -cycle detection.

## Part 3: 1-VASS

Coverability in 1-VASS can be decided in  $n^2$  time

Coverability in (binary) 1-VASS is in  $AC^1$ .

### Open Problem

What is the *exact* complexity of coverability in 1-VASS?

# Preview of this talk

## Part 1: $d$ -VASS

Coverability in  $d$ -VASS can be decided in  $n^{2^{\mathcal{O}(d)}}$  time.

Coverability in  $d$ -VASS conditionally requires  $n^{2^{\Omega(d)}}$  time.

Parameterised reduction from  $k$ -clique detection.

## Part 2: 2-VASS

Coverability in 2-VASS can be decided in  $n^C$  time.

Coverability in 2-VASS conditionally requires  $n^2$  time.

Linear-time reduction from  $k$ -cycle detection.

## Part 3: 1-VASS

Coverability in 1-VASS can be decided in  $n^2$  time

Coverability in (binary) 1-VASS is in  $AC^1$ .

### Open Problem

What is the *exact* complexity of coverability in 1-VASS?

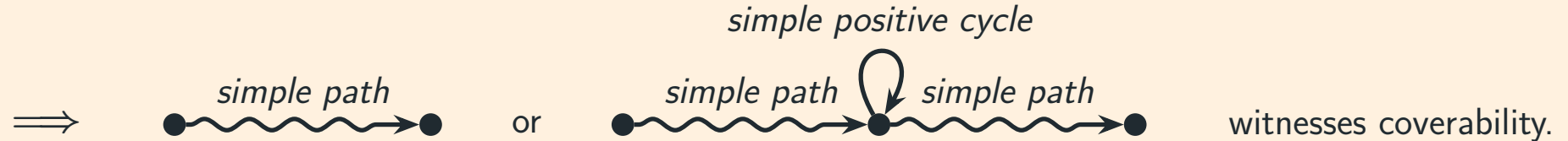
# Complexity of Coverability in 1-VASS

**Theorem.** Coverability in 1-VASS can be decided in  $\mathcal{O}(n^2)$  time.

# Complexity of Coverability in 1-VASS

**Theorem.** Coverability in 1-VASS can be decided in  $\mathcal{O}(n^2)$  time.

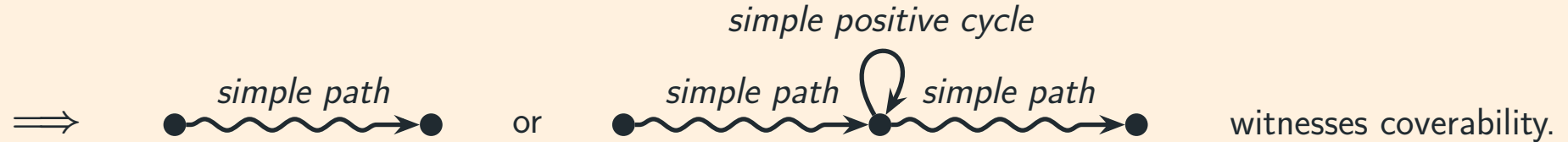
**Observation.** One need not take non-positive weight cycles.



# Complexity of Coverability in 1-VASS

**Theorem.** Coverability in 1-VASS can be decided in  $\mathcal{O}(n^2)$  time.

**Observation.** One need not take non-positive weight cycles.

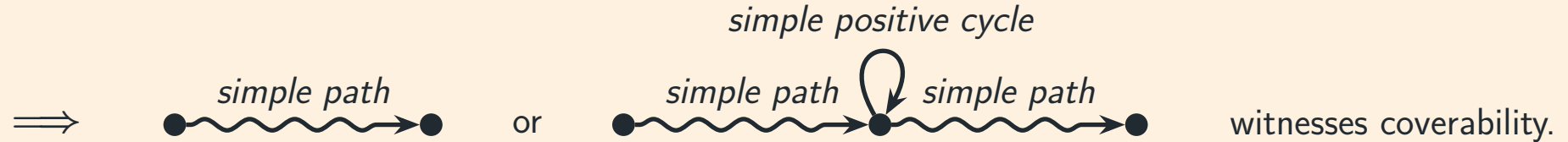


**Idea.** Modify the Bellman-Ford  $\mathcal{O}(|V||E|)$ -time algorithm for shortest paths in integer-weighted graphs.

# Complexity of Coverability in 1-VASS

**Theorem.** Coverability in 1-VASS can be decided in  $\mathcal{O}(n^2)$  time.

**Observation.** One need not take non-positive weight cycles.



**Idea.** Modify the Bellman-Ford  $\mathcal{O}(|V||E|)$ -time algorithm for shortest paths in integer-weighted graphs.

**Theorem.** Coverability in (binary-encoded) 1-VASS is in  $AC^1$ . [Shakiba, S-B, and Zetsche '25]

This matches the state-of-the-art for shortest paths in  $\mathbb{N}$ -weighted directed graphs. [Cook '85]

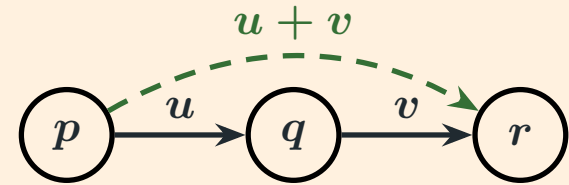
# An Efficient Algorithm for Coverability in 1-VASS

STEP 0: Let  $\mathcal{V}_0$  be the given 1-VASS.

# An Efficient Algorithm for Coverability in 1-VASS

STEP 0: Let  $\mathcal{V}_0$  be the given 1-VASS.

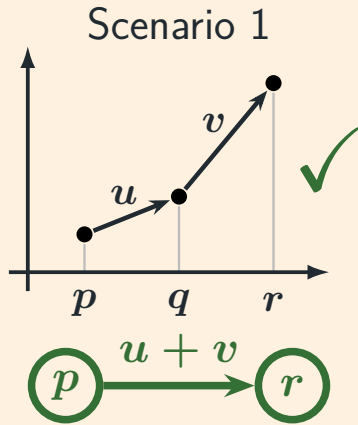
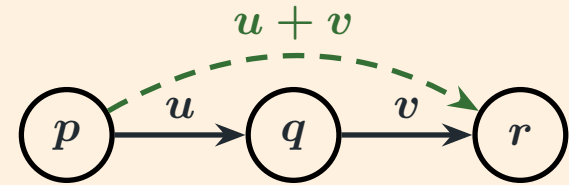
STEP 1: Create  $\mathcal{V}_1$  from  $\mathcal{V}_0$  by adding “shortcut transitions”:



# An Efficient Algorithm for Coverability in 1-VASS

STEP 0: Let  $\mathcal{V}_0$  be the given 1-VASS.

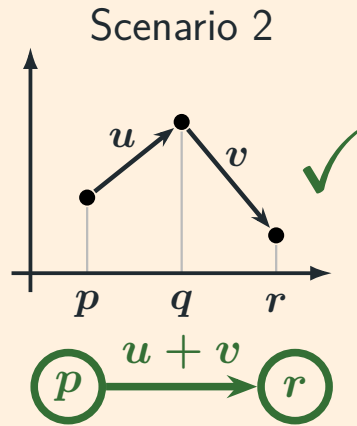
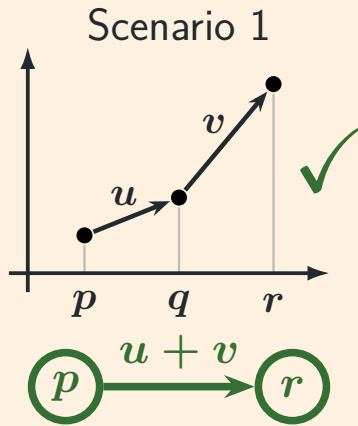
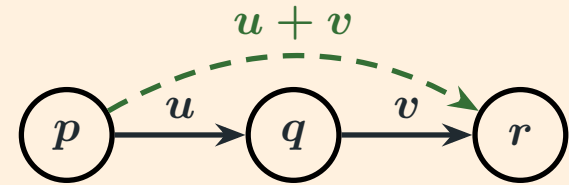
STEP 1: Create  $\mathcal{V}_1$  from  $\mathcal{V}_0$  by adding “shortcut transitions”:



# An Efficient Algorithm for Coverability in 1-VASS

STEP 0: Let  $\mathcal{V}_0$  be the given 1-VASS.

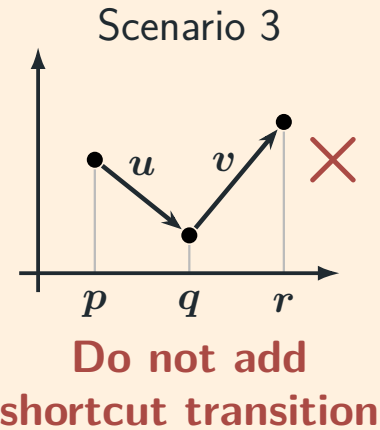
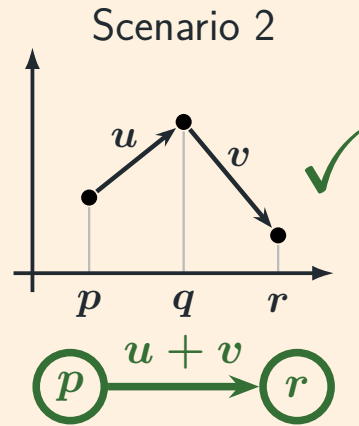
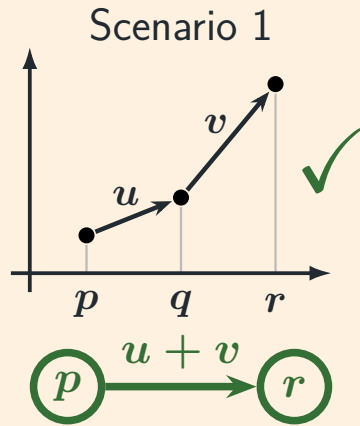
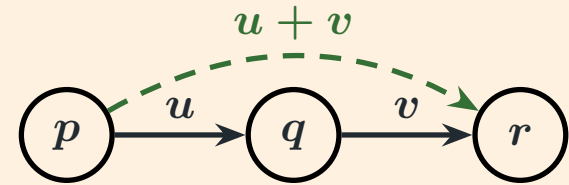
STEP 1: Create  $\mathcal{V}_1$  from  $\mathcal{V}_0$  by adding “shortcut transitions”:



# An Efficient Algorithm for Coverability in 1-VASS

STEP 0: Let  $\mathcal{V}_0$  be the given 1-VASS.

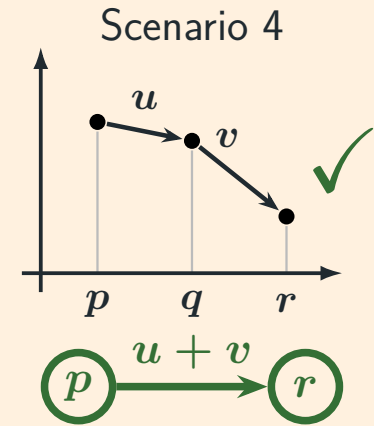
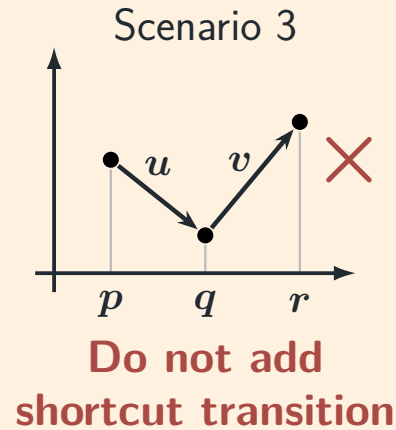
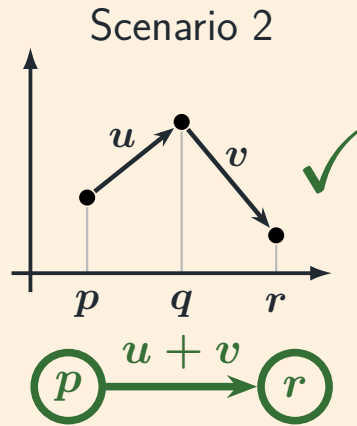
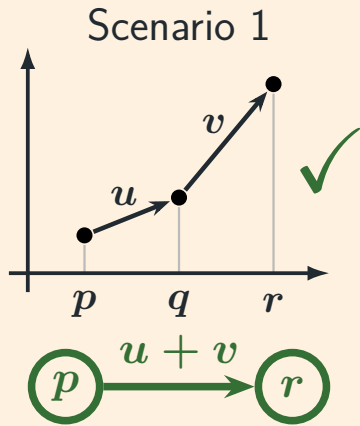
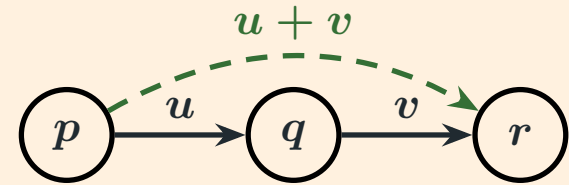
STEP 1: Create  $\mathcal{V}_1$  from  $\mathcal{V}_0$  by adding “shortcut transitions”:



# An Efficient Algorithm for Coverability in 1-VASS

STEP 0: Let  $\mathcal{V}_0$  be the given 1-VASS.

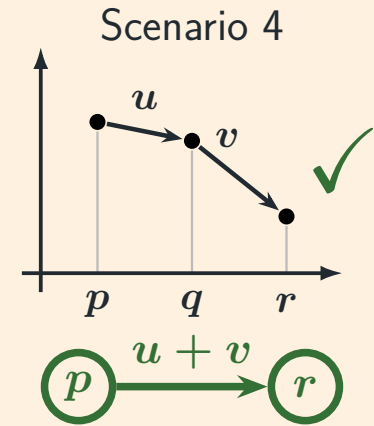
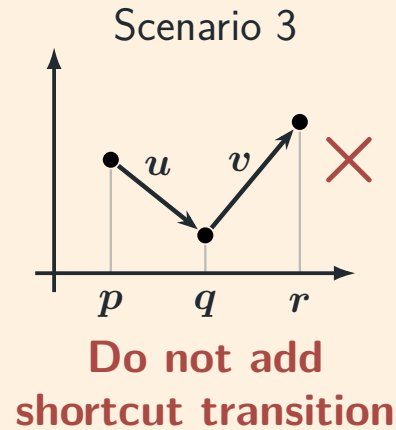
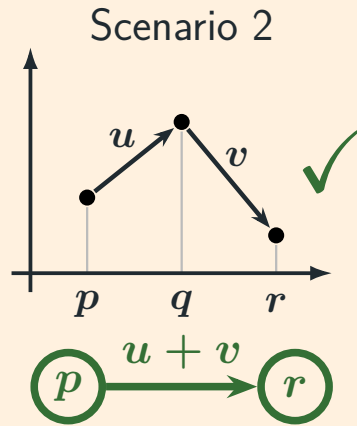
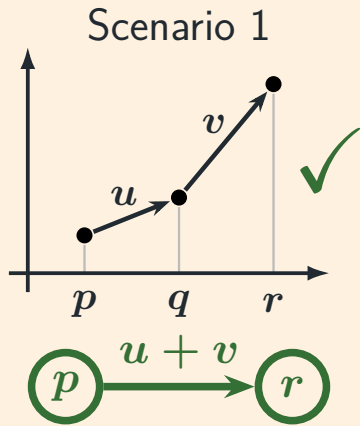
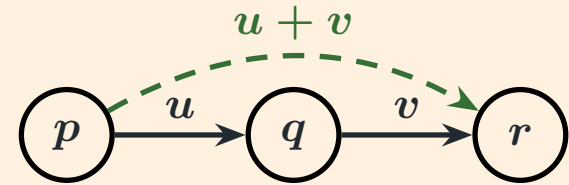
STEP 1: Create  $\mathcal{V}_1$  from  $\mathcal{V}_0$  by adding “shortcut transitions”:



# An Efficient Algorithm for Coverability in 1-VASS

STEP 0: Let  $\mathcal{V}_0$  be the given 1-VASS.

STEP 1: Create  $\mathcal{V}_1$  from  $\mathcal{V}_0$  by adding “shortcut transitions”:

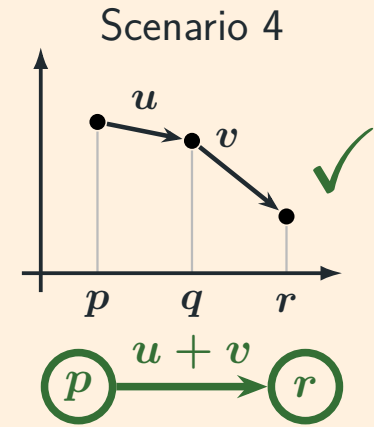
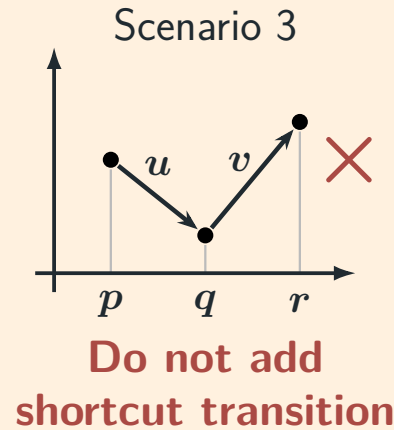
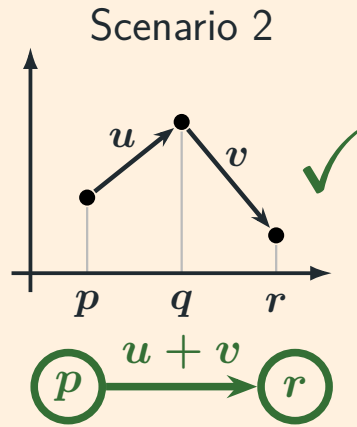
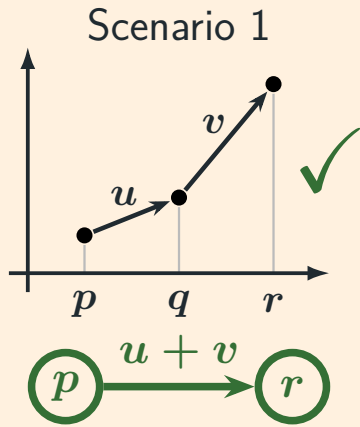
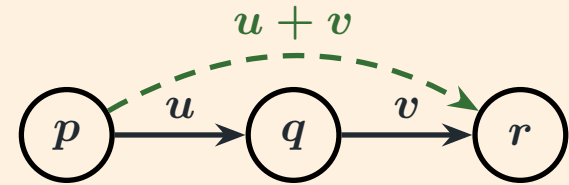


STEP i: Repeat  $k = 2 \lceil \log n \rceil$  many times.

# An Efficient Algorithm for Coverability in 1-VASS

STEP 0: Let  $\mathcal{V}_0$  be the given 1-VASS.

STEP 1: Create  $\mathcal{V}_1$  from  $\mathcal{V}_0$  by adding “shortcut transitions”:



STEP  $i$ : Repeat  $k = 2 \lceil \log n \rceil$  many times.

**Claim.** There is a covering run in  $\mathcal{V}_0$  if and only if there is a covering run of **length**  $\leq 2$  in  $\mathcal{V}_k$ .

# What is the Fine-Grained Complexity of Coverability in 1-VASS?

**Theorem.** Coverability in 1-VASS can be decided in  $\mathcal{O}(n^2)$  time.

**Observation.** One need not take non-positive weight cycles.

**Open Problem.** Does there exist a  $o(n^2)$ -time algorithm for coverability in 1-VASS?



**Idea.** Modify the  $\mathcal{O}(|V||E|)$  Bellman-Ford algorithm for shortest paths in integer-weighted graphs.

**Theorem.** Coverability in (binary-encoded) 1-VASS is in  $AC^1$ . [Shakiba, S-B, and Zetsche '25]

This matches the state-of-the-art for shortest paths in  $\mathbb{N}$ -weighted directed graphs. [Cook '85]

# What is the Fine-Grained Complexity of Coverability in 1-VASS?

**Theorem.** Coverability in 1-VASS can be decided in  $\mathcal{O}(n^2)$  time.

**Observation.** One need not take non-positive weight cycles.

**Open Problem.** Does there exist a  $o(n^2)$ -time algorithm for coverability in 1-VASS?

lift ideas?

new ideas?

**Theorem.** There is a *randomised* algorithm which computes a shortest path tree from  $s$  or finds a negative cycle in  $n$  polylog( $n$ ) time.

[Bernstein, Nanongkai, and Wulff-Nilsen '22]

[Bringmann, Cassis, and Fischer '23]

**Theorem.** Coverability in (binary-encoded) 1-VASS is in  $AC^1$ .

[Shakiba, S-B, and Zetsche '25]

This matches the state-of-the-art for shortest paths in  $\mathbb{N}$ -weighted directed graphs.

[Cook '85]

# Precise Complexity Analysis of Coverability in Vector Addition Systems

**Theorem.** Assuming the exponential time hypothesis, coverability in  $d$ -VASS requires  $n^{2^{\Omega(d)}}$  time.

**Idea.** Reduce detecting a  $2^d$ -clique in a  $2^d$ -partite  $n$ -vertex graph to coverability in  $d$ -VASS.

**Theorem.** Assuming the  $k$ -cycle hypothesis, coverability in 2-VASS “requires quadratic time”.

**Idea.** Reduce, in linear time,  $k$ -cycle detection in  $m$ -edge directed graphs to coverability in 2-VASS.

**Theorem.** Coverability in 1-VASS can be decided in  $\mathcal{O}(n^2)$  time and is in  $AC^1$ .

**Open problem.** Does there exist an  $o(n^2)$ -time algorithm for coverability in 1-VASS?

## Thank You!



Presented by Henry Sinclair-Banks, MPI-SWS, Kaiserslautern, Germany 

INFINITY 2026, Royal Holloway, University of London, Egham, UK 

Presentation made  
with BeamerikZ