Attacking the General Isogeny Problem

Maria Corte-Real Santos
University College London

Based on joint work with Craig Costello, Sam Frengley and Jia Shi

The Isogeny Club
October 11, 2022

Outline

- The General Isogeny Problem in Dimensions 1 and 2

Outline

- The General Isogeny Problem in Dimensions 1 and 2
- Viewing this as a path finding problem

Outline

- The General Isogeny Problem in Dimensions 1 and 2
- Viewing this as a path finding problem
- Strategies for finding paths in a graph

Outline

- The General Isogeny Problem in Dimensions 1 and 2
- Viewing this as a path finding problem
- Strategies for finding paths in a graph
- Application to Dimension 1

Outline

- The General Isogeny Problem in Dimensions 1 and 2
- Viewing this as a path finding problem
- Strategies for finding paths in a graph
- Application to Dimension 1
- Application to Dimension 2

Outline

- The General Isogeny Problem in Dimensions 1 and 2
- Viewing this as a path finding problem
- Strategies for finding paths in a graph
- Application to Dimension 1
- Application to Dimension 2
- Results

Elliptic Curves and j-invariants

We consider *supersingular* elliptic curves $E/\bar{\mathbb{F}}_p$ ($p > 3$).

Elliptic Curves and j-invariants

We consider *supersingular* elliptic curves $E/\bar{\mathbb{F}}_p$ $(p > 3)$.

We are interested in elliptic curves up to $\bar{\mathbb{F}}_p$-isomorphism, and so we label each class by the j-invariants.

Definition

Let $E : y^2 = x^3 + ax + b$. Then, the j-invariant of E is

$$j(E) = 1728\frac{4a^3}{4a^3 + 27b^2}.$$

Every supersingular E admits a model over $\mathbb{F}_{p^2}$ and $j(E) \in \mathbb{F}_{p^2}$.

Elliptic Curves and j-invariants

We consider *supersingular* elliptic curves $E/\bar{\mathbb{F}}_p$ ($p > 3$).

We are interested in elliptic curves up to $\bar{\mathbb{F}}_p$-isomorphism, and so we label each class by the j-invariants.

Definition

Let $E : y^2 = x^3 + ax + b$. Then, the j-invariant of E is

$$j(E) = 1728\frac{4a^3}{4a^3 + 27b^2}.$$

Every supersingular E admits a model over $\mathbb{F}_{p^2}$ and $j(E) \in \mathbb{F}_{p^2}$.

We consider N-isogenies between them: isogenies with kernel $\mathbb{Z}/N\mathbb{Z}$ generated by a point $P \in E(\bar{\mathbb{F}}_p)$.

The General Isogeny Problem in One Dimension

In its most general form, the *supersingular isogeny problem* asks to find an isogeny

$$\phi : E_1 \longrightarrow E_2,$$

between two given supersingular elliptic curves $E_1/\mathbb{F}_{p^2}$ and $E_2/\mathbb{F}_{p^2}$.

The General Isogeny Problem in One Dimension

In its most general form, the *supersingular isogeny problem* asks to find an isogeny

$$\phi : E_1 \longrightarrow E_2,$$

between two given supersingular elliptic curves $E_1/\mathbb{F}_{p^2}$ and $E_2/\mathbb{F}_{p^2}$.

Note, we do not assume knowledge of:

- torsion point information
- degree of the isogeny

Generalising to Two Dimensions

To generalise supersingular elliptic curves over $\bar{\mathbb{F}}_p$ ($p > 3$) to dimension 2, we consider *superspecial (principally polarised) abelian surfaces* over $\bar{\mathbb{F}}_p$.

Generalising to Two Dimensions

To generalise supersingular elliptic curves over $\bar{\mathbb{F}}_p$ ($p > 3$) to dimension 2, we consider *superspecial (principally polarised) abelian surfaces* over $\bar{\mathbb{F}}_p$.

There are two types:

Generalising to Two Dimensions

To generalise supersingular elliptic curves over $\bar{\mathbb{F}}_p$ ($p > 3$) to dimension 2, we consider *superspecial (principally polarised) abelian surfaces* over $\bar{\mathbb{F}}_p$.

There are two types:

1. Products of supersingular elliptic curves $E \times E'$

Generalising to Two Dimensions

To generalise supersingular elliptic curves over $\overline{\mathbb{F}}_p$ ($p > 3$) to dimension 2, we consider *superspecial (principally polarised) abelian surfaces* over $\overline{\mathbb{F}}_p$.

There are two types:

1. Products of supersingular elliptic curves $E \times E'$
2. Jacobians $\mathrm{Jac}(C)$ of genus 2 curves C

Generalising to Two Dimensions

To generalise supersingular elliptic curves over $\overline{\mathbb{F}}_p$ ($p > 3$) to dimension 2, we consider *superspecial (principally polarised) abelian surfaces* over $\overline{\mathbb{F}}_p$.

There are two types:

1. Products of supersingular elliptic curves $E \times E'$
2. Jacobians $\mathrm{Jac}(C)$ of genus 2 curves C

We consider them up to $\overline{\mathbb{F}}_p$-isomorphism and label these classes with:

Generalising to Two Dimensions

To generalise supersingular elliptic curves over $\overline{\mathbb{F}}_p$ ($p > 3$) to dimension 2, we consider *superspecial (principally polarised) abelian surfaces* over $\overline{\mathbb{F}}_p$.

There are two types:

1. Products of supersingular elliptic curves $E \times E'$
2. Jacobians $\mathrm{Jac}(C)$ of genus 2 curves C

We consider them up to $\overline{\mathbb{F}}_p$-isomorphism and label these classes with:

1. Sets of j-invariants $\{ j(E),\ j(E') \}$

Generalising to Two Dimensions

To generalise supersingular elliptic curves over $\overline{\mathbb{F}}_p$ ($p > 3$) to dimension 2, we consider *superspecial (principally polarised) abelian surfaces* over $\overline{\mathbb{F}}_p$.

There are two types:

1. Products of supersingular elliptic curves $E \times E'$
2. Jacobians $\mathrm{Jac}(C)$ of genus 2 curves C

We consider them up to $\overline{\mathbb{F}}_p$-isomorphism and label these classes with:

1. Sets of j-invariants $\{ j(E),\ j(E') \}$
2. Igusa–Clebsch invariants $I_2(C), I_4(C), I_6(C), I_{10}(C)$ (lie in weighted projective space $\mathbb{P}(2, 4, 6, 10)$, subscript denotes the weight).

Generalising to Two Dimensions

To generalise supersingular elliptic curves over $\bar{\mathbb{F}}_p$ ($p > 3$) to dimension 2, we consider *superspecial (principally polarised) abelian surfaces* over $\bar{\mathbb{F}}_p$.

There are two types:

1. Products of supersingular elliptic curves $E \times E'$
2. Jacobians $\mathrm{Jac}(C)$ of genus 2 curves C

We consider them up to $\bar{\mathbb{F}}_p$-isomorphism and label these classes with:

1. Sets of j-invariants $\{ j(E),\ j(E') \}$
2. Igusa–Clebsch invariants $I_2(C), I_4(C), I_6(C), I_{10}(C)$ (lie in weighted projective space $\mathbb{P}(2, 4, 6, 10)$, subscript denotes the weight).

Every superspecial p.p. abelian surface admits a model over $\mathbb{F}_{p^2}$ and its invariants lie in $\mathbb{F}_{p^2}$.

(N, N)-Isogenies

An (N, N)-isogeny is an isogeny[1] $\phi\colon A \longrightarrow A'$, between p.p. abelian surfaces A, A' where:

- $\ker \phi \cong (\mathbb{Z}/N\mathbb{Z})^2$; and
- $\ker \phi$ is maximal isotropic with respect to the N-Weil Pairing, i.e.

$$\forall P, Q \in \ker \phi : \ e_N(P, Q) = 1.$$

This ensures A' comes equipped with a principal polarisation.

[1] i.e., surjective group homomorphism with finite kernel

(N, N)-Isogenies

An (N, N)-isogeny is an isogeny[1] $\phi\colon A \longrightarrow A'$, between p.p. abelian surfaces A, A' where:

- $\ker\phi \cong (\mathbb{Z}/N\mathbb{Z})^2$; and
- $\ker\phi$ is maximal isotropic with respect to the N-Weil Pairing, i.e.

$$\forall P, Q \in \ker\phi : \ e_N(P, Q) = 1.$$

This ensures A' comes equipped with a principal polarisation.

There are

$$N^3 \prod_{\substack{\text{primes} \\ \ell | N}} \frac{1}{\ell^3}(\ell + 1)(\ell^2 + 1)$$

(N, N)-isogenies from a surface A.

[1] i.e., surjective group homomorphism with finite kernel

General Isogeny Problem in Two Dimensions

In its most general form, the superspecial isogeny problem in two dimensions asks to find an isogeny

$$\phi\colon A \longrightarrow A',$$

between two superspecial (p.p.) abelian surfaces $A/\mathbb{F}_{p^2}$ and $A'/\mathbb{F}_{p^2}$.

General Isogeny Problem in Two Dimensions

In its most general form, the superspecial isogeny problem in two dimensions asks to find an isogeny

$$\phi\colon A \longrightarrow A',$$

between two superspecial (p.p.) abelian surfaces $A/\mathbb{F}_{p^2}$ and $A'/\mathbb{F}_{p^2}$.

Again, we do not assume any extra information.

Reframing as a Path Finding Problem

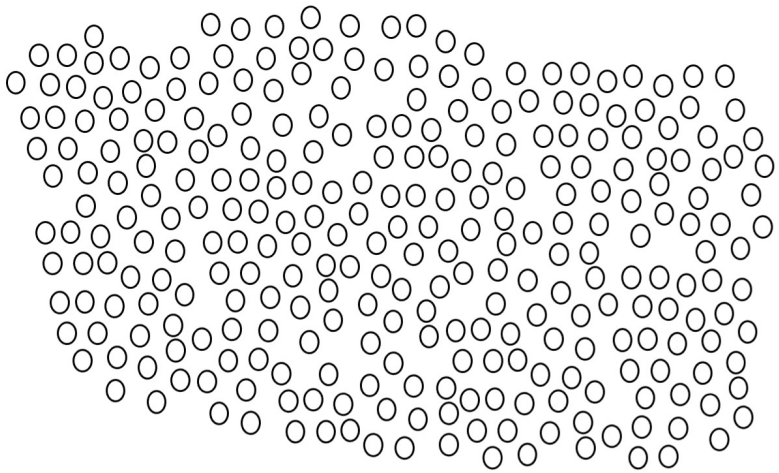
The general isogeny problem in both dimensions can be viewed as finding a path between two nodes in a related graph.

Reframing as a Path Finding Problem

The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One

Reframing as a Path Finding Problem

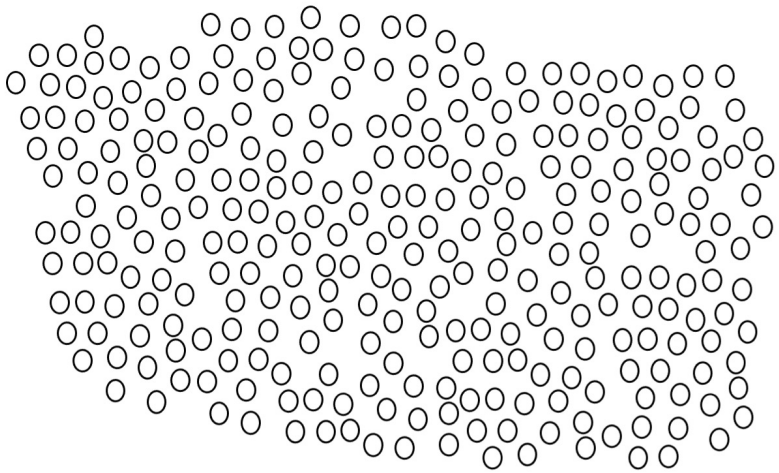
The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One



Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of supersingular E, represented by the j-invariant
Edges: N-isogenies

Reframing as a Path Finding Problem

The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One



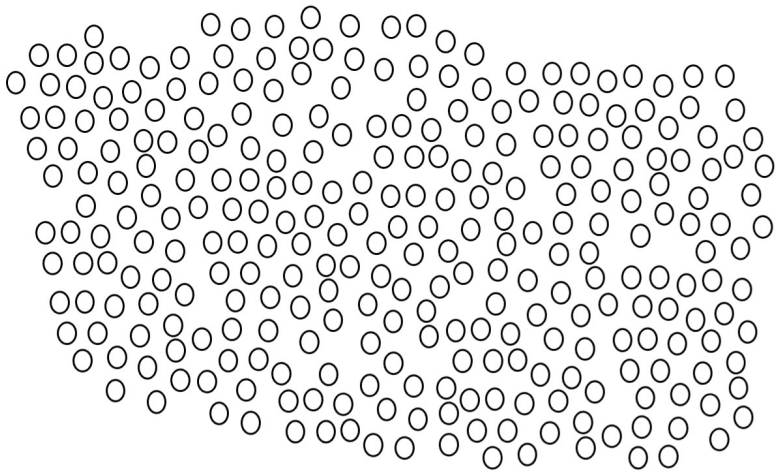
Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of supersingular E, represented by the j-invariant
Edges: N-isogenies

Properties:

- There are $\approx \frac{p}{12}$ vertices
- A random walk of $\log(p)$ steps is almost as good as uniformly sampling vertices (Ramanujan property)
- Path finding is conjectured to be hard for classical and quantum computers

Reframing as a Path Finding Problem

The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One



Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of supersingular E, represented by the j-invariant
Edges: N-isogenies

Properties:

- There are $\approx \frac{p}{12}$ vertices
- A random walk of $\log(p)$ steps is almost as good as uniformly sampling vertices (Ramanujan property)
- Path finding is conjectured to be hard for classical and quantum computers

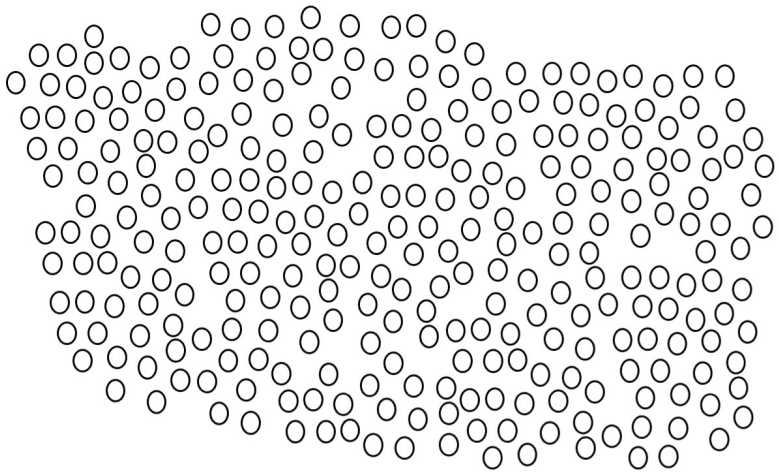
Finding a path between two nodes j_1, j_2 = Finding an isogeny between E_1, E_2

Reframing as a Path Finding Problem

The (N, N)-Isogeny Graph $\Gamma_2(N; p)$ in Dimension Two

Reframing as a Path Finding Problem

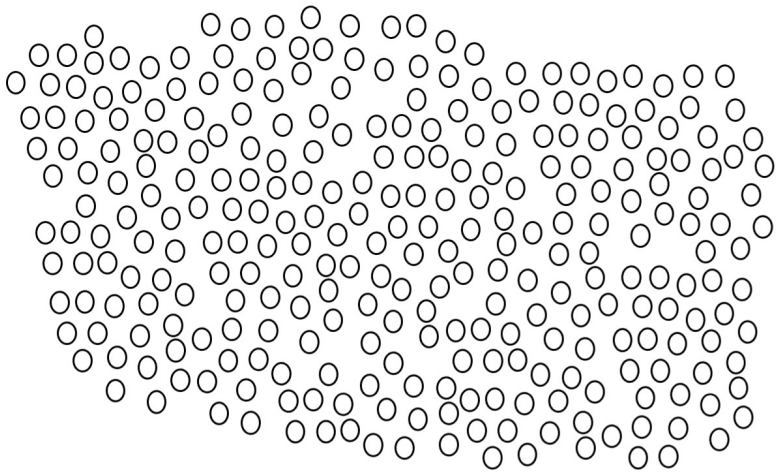
The (N, N)-Isogeny Graph $\Gamma_2(N; p)$ in Dimension Two



Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of abelian surfaces A
Edges: (N, N)-isogenies

Reframing as a Path Finding Problem

The (N, N)-Isogeny Graph $\Gamma_2(N; p)$ in Dimension Two



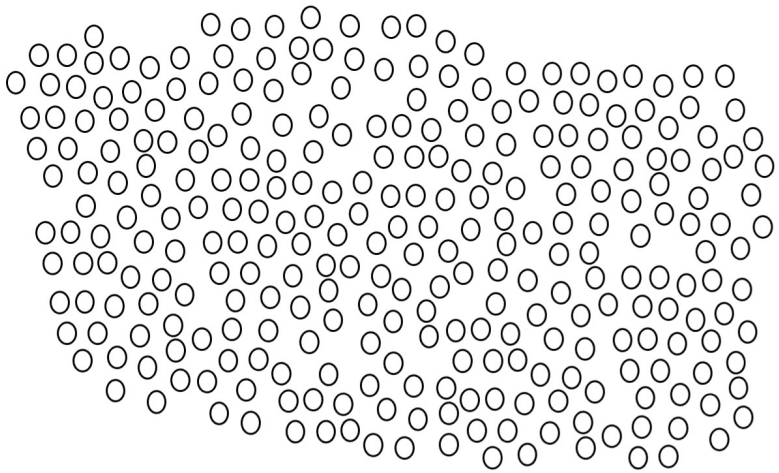
Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of abelian surfaces A
Edges: (N, N)-isogenies

Properties:

- There are $O(p^3)$ vertices
- Ramanujan? Not quite, but expansion is *'good enough'*
- Path finding still conjectured to be hard, but hardness properties more *unknown*

Reframing as a Path Finding Problem

The (N, N)-Isogeny Graph $\Gamma_2(N; p)$ in Dimension Two



Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of abelian surfaces A
Edges: (N, N)-isogenies

Properties:

- There are $O(p^3)$ vertices
- Ramanujan? Not quite, but expansion is *'good enough'*
- Path finding still conjectured to be hard, but hardness properties more *unknown*

Finding path between nodes A_1, A_2 = Finding an isogeny between A_1, A_2

Attacking Path Finding Problems

A first attempt would be to walk in the graph from the *start node* until reaching the *end node*.

Attacking Path Finding Problems

A first attempt would be to walk in the graph from the *start node* until reaching the *end node*.

As the graphs will be finite, this procedure will terminate, but likely not *efficiently*.

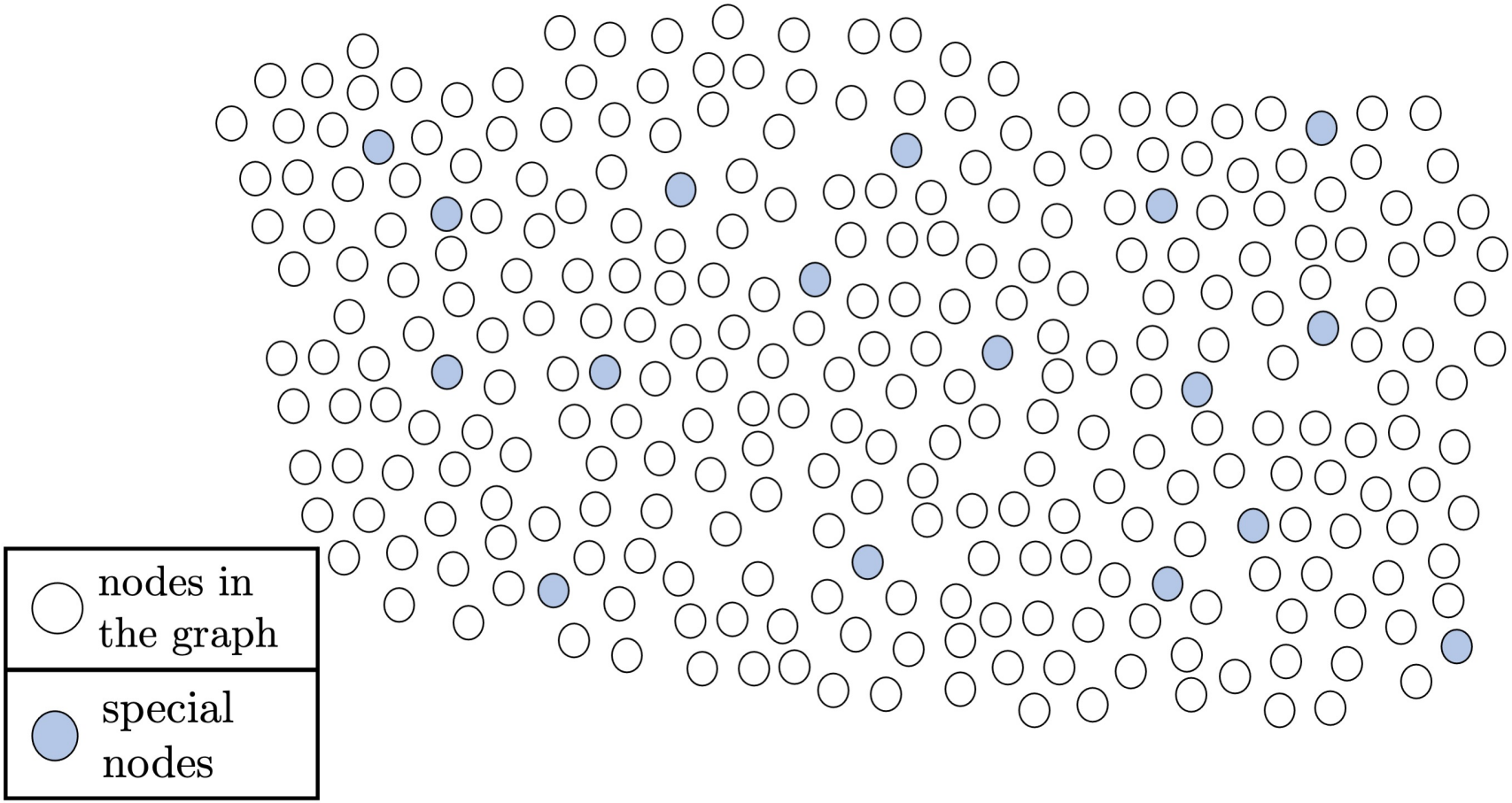
Attacking Path Finding Problems

A first attempt would be to walk in the graph from the *start node* until reaching the *end node*.

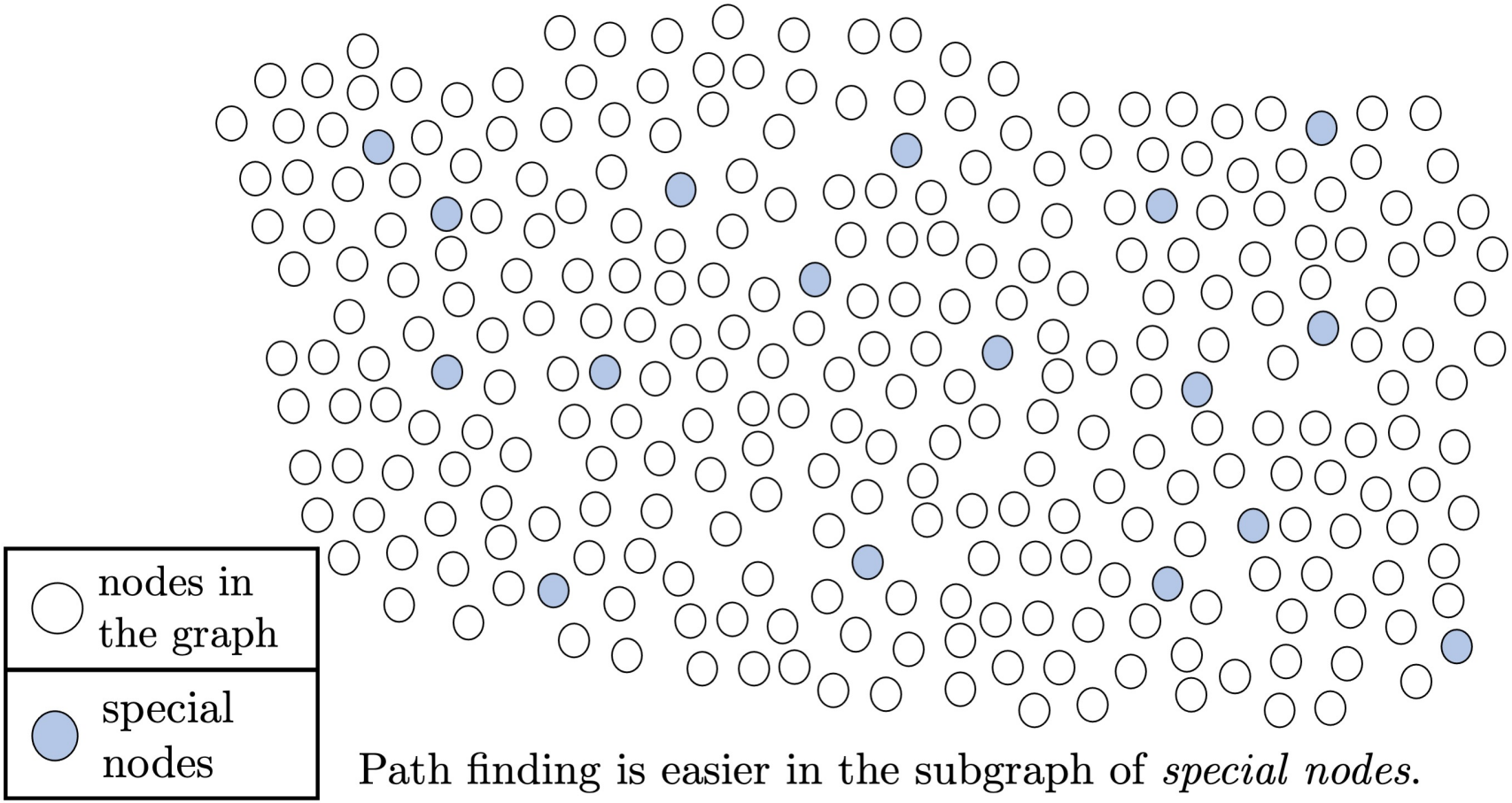
As the graphs will be finite, this procedure will terminate, but likely not *efficiently*.

Can we find a better method (excluding the MitM approach)?

Reducing to Easier Problems using Special Subsets

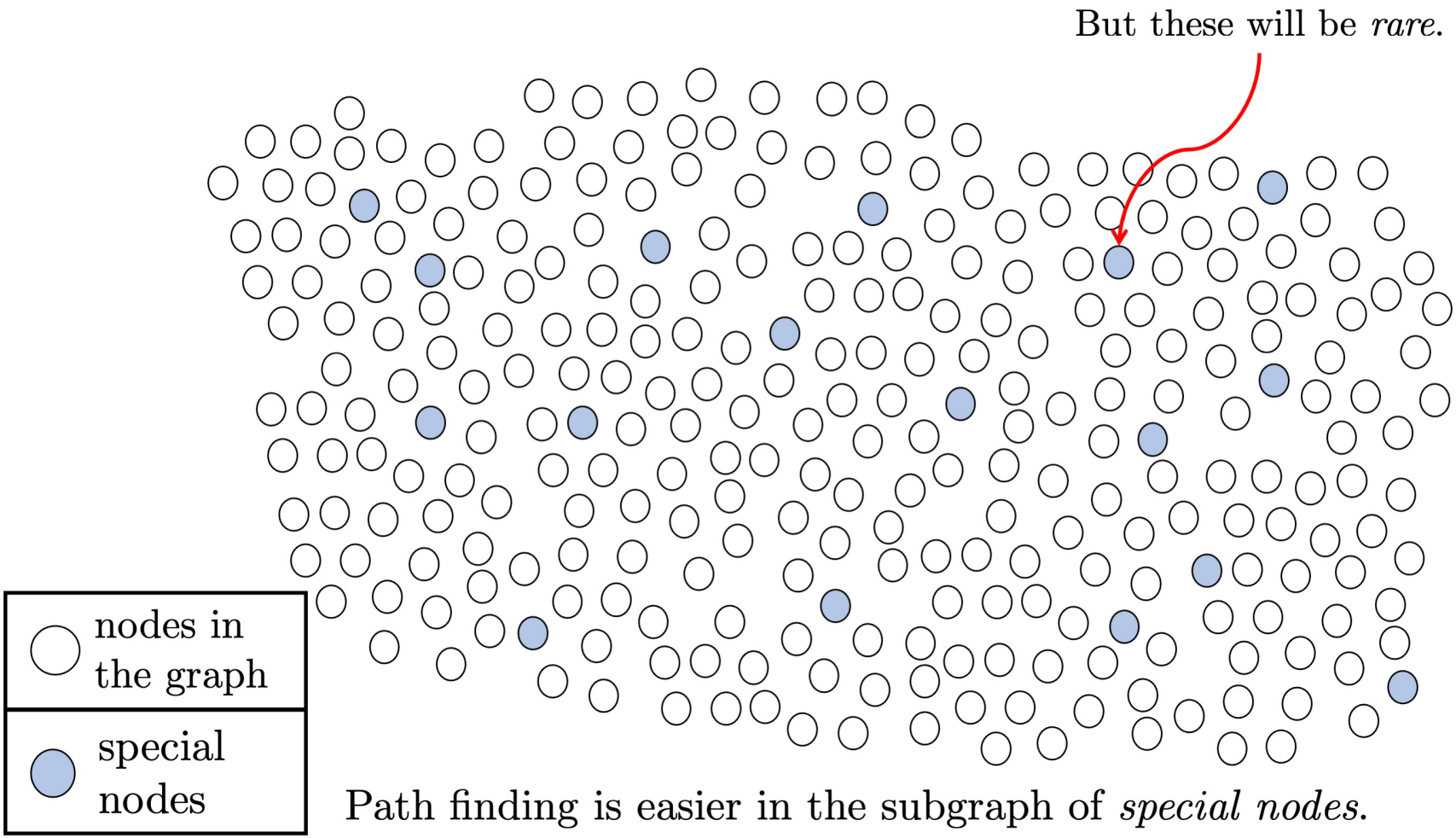


Reducing to Easier Problems using Special Subsets



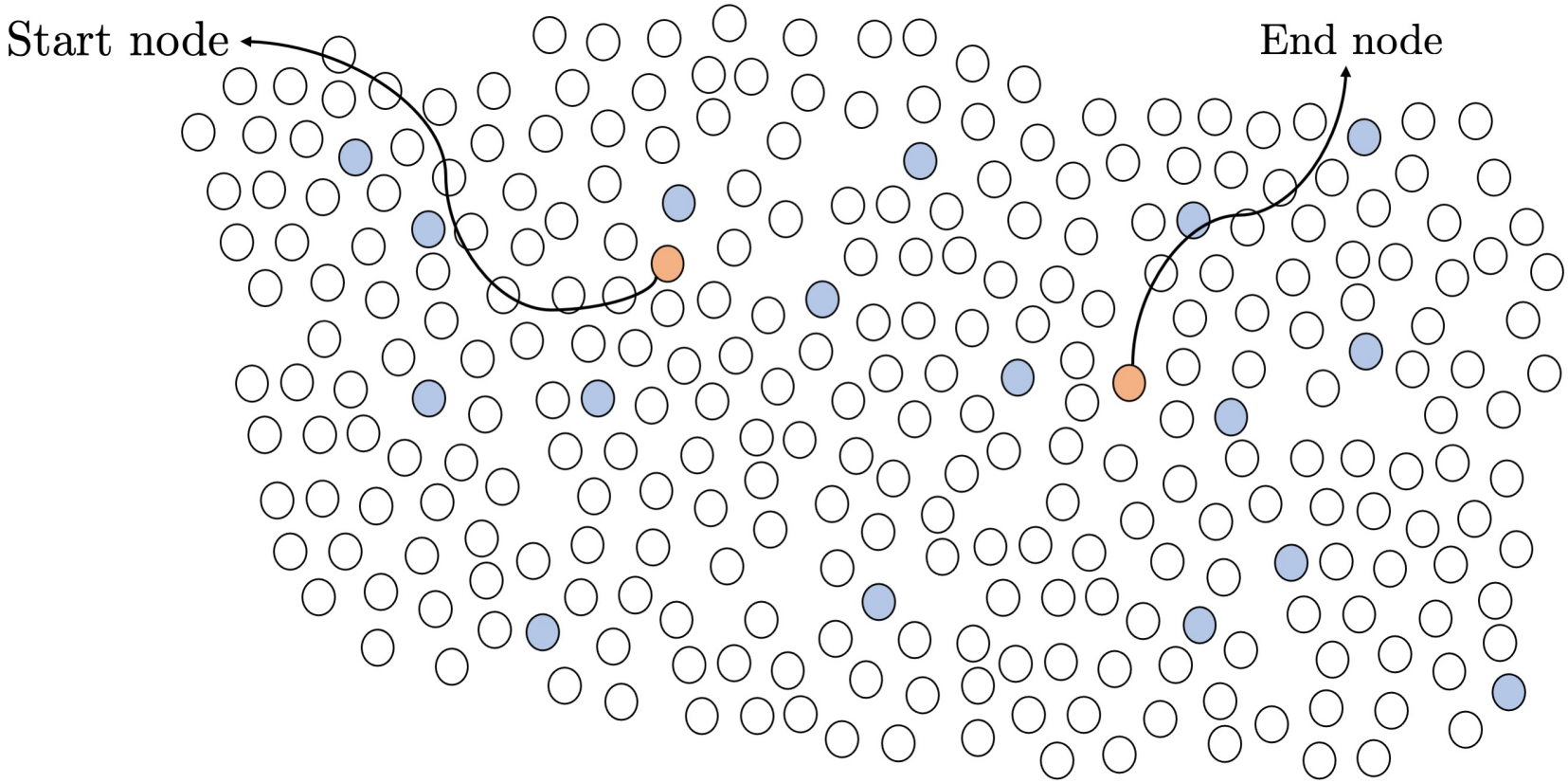
Path finding is easier in the subgraph of *special nodes*.

Reducing to Easier Problems using Special Subsets

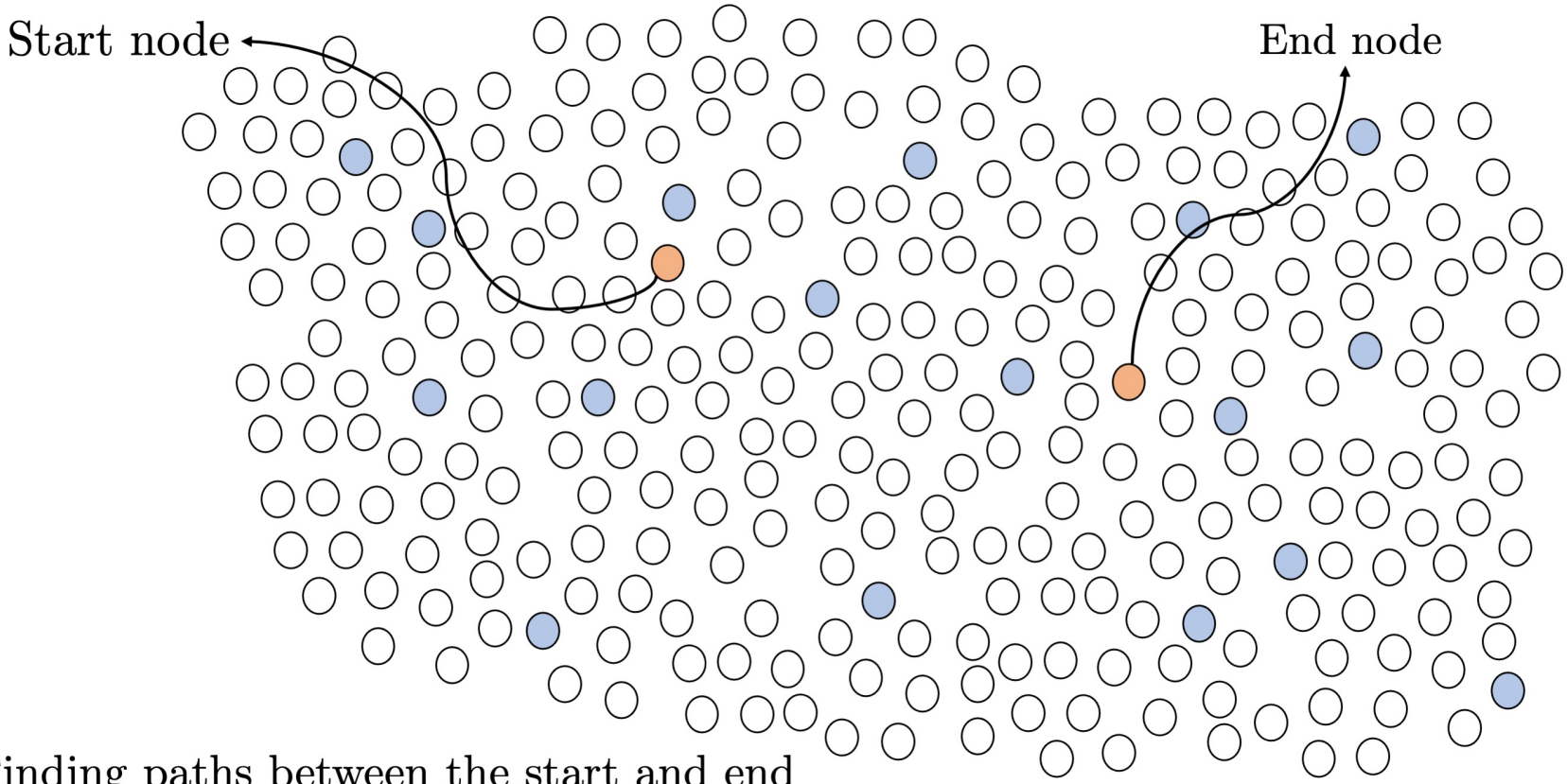


Path finding is easier in the subgraph of *special nodes*.

Reducing to Easier Problems using Special Subsets

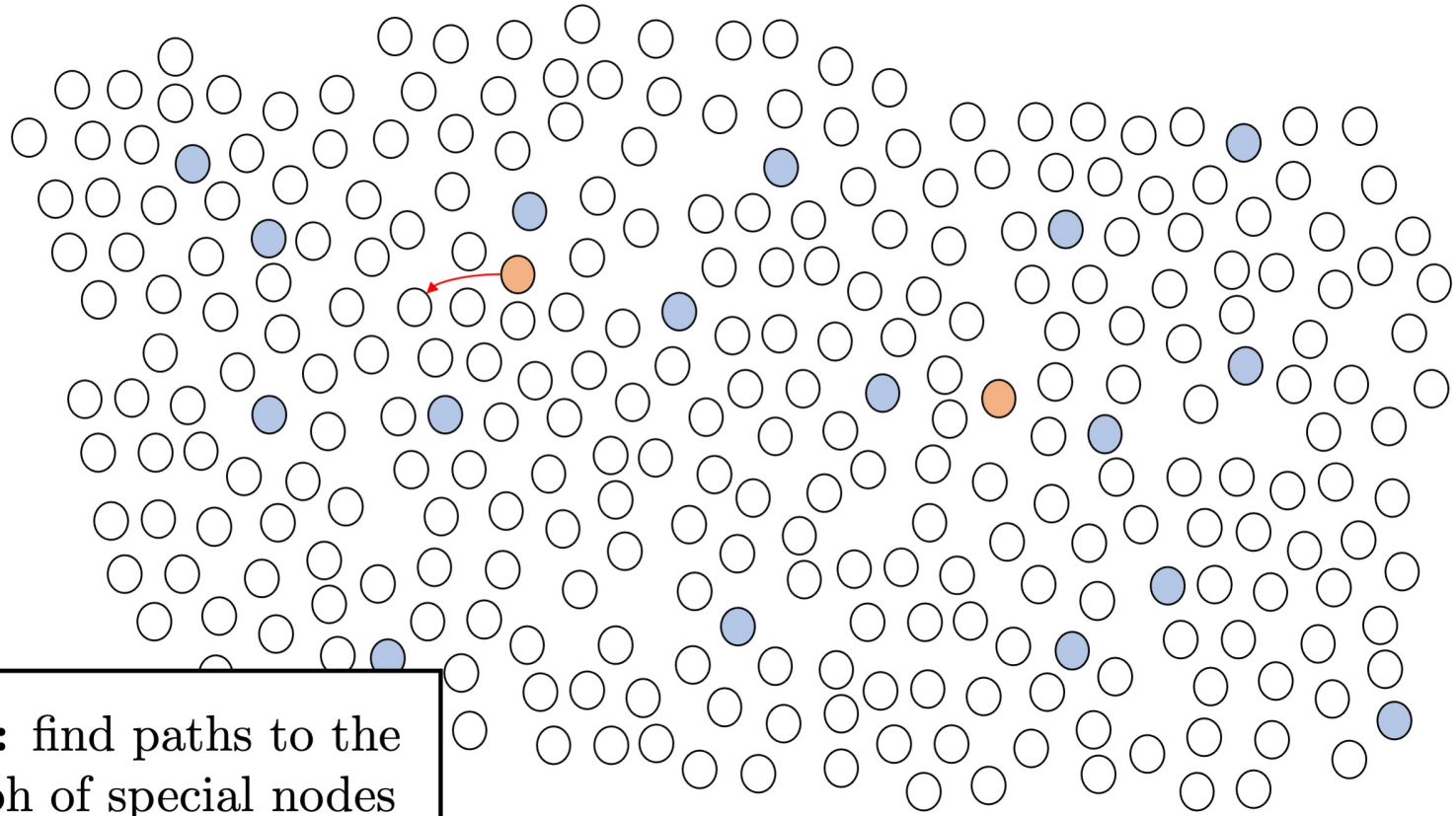


Reducing to Easier Problems using Special Subsets



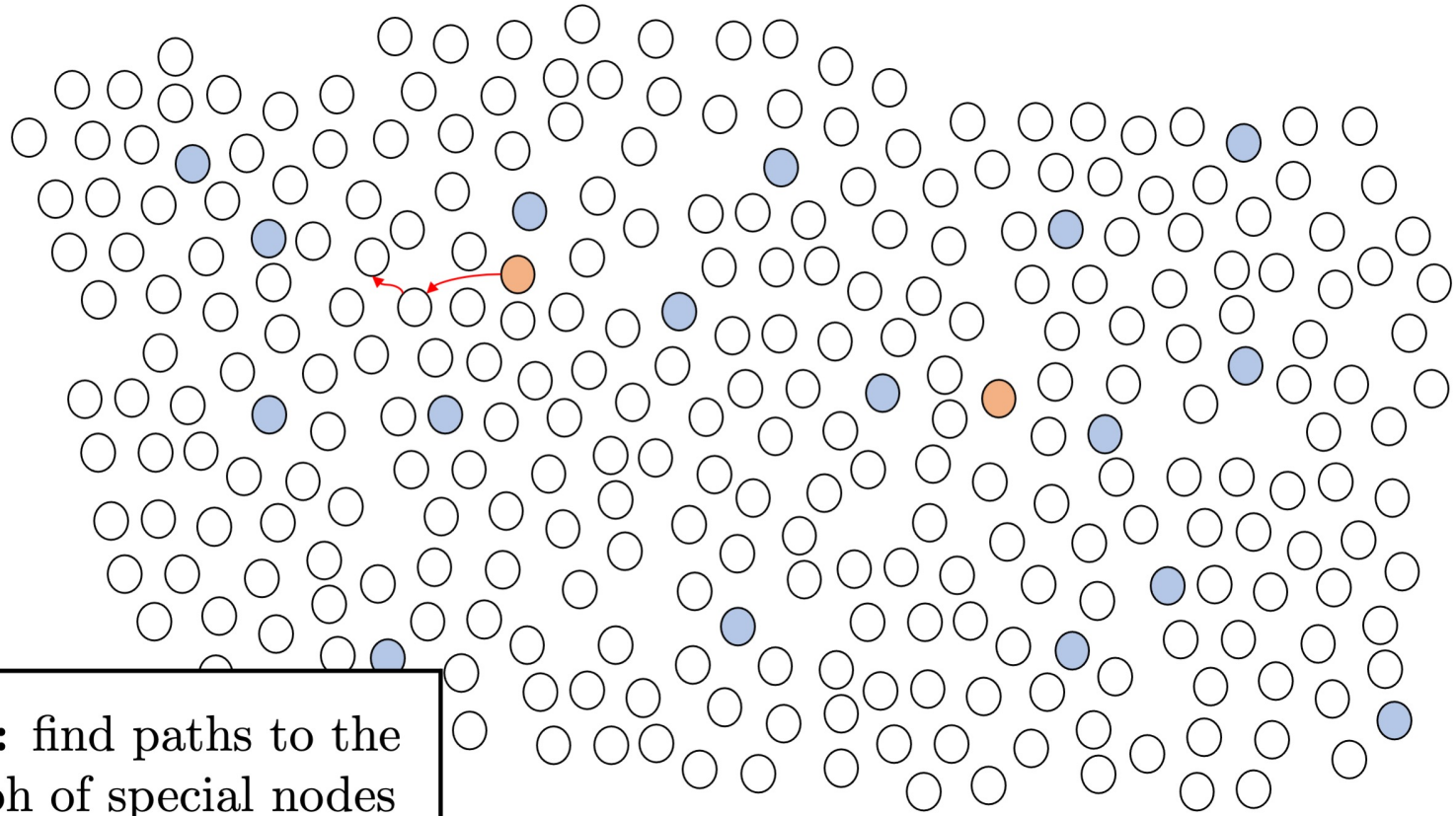
Finding paths between the start and end node can be split up into two steps.

Reducing to Easier Problems using Special Subsets



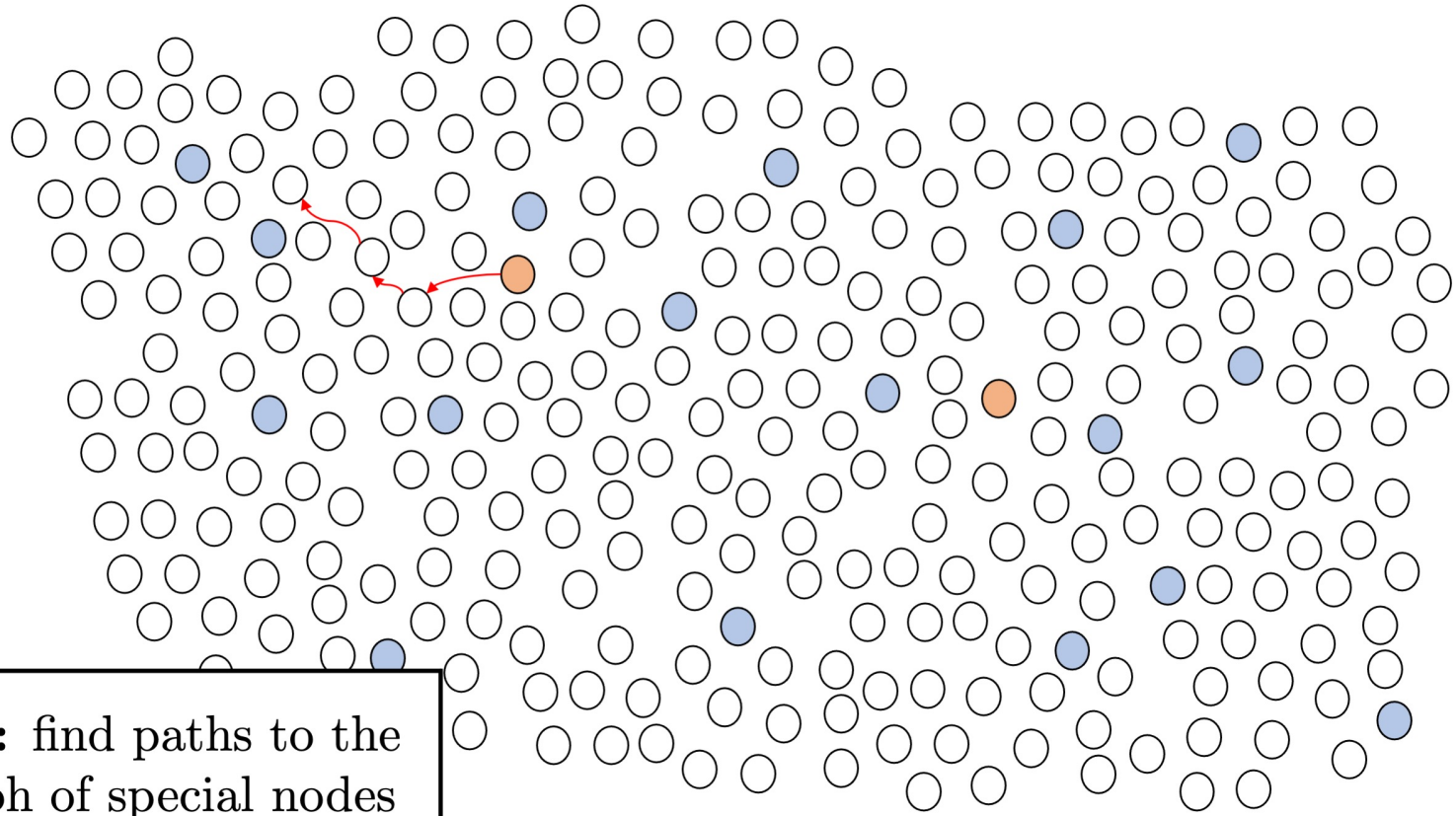
Step 1: find paths to the subgraph of special nodes

Reducing to Easier Problems using Special Subsets



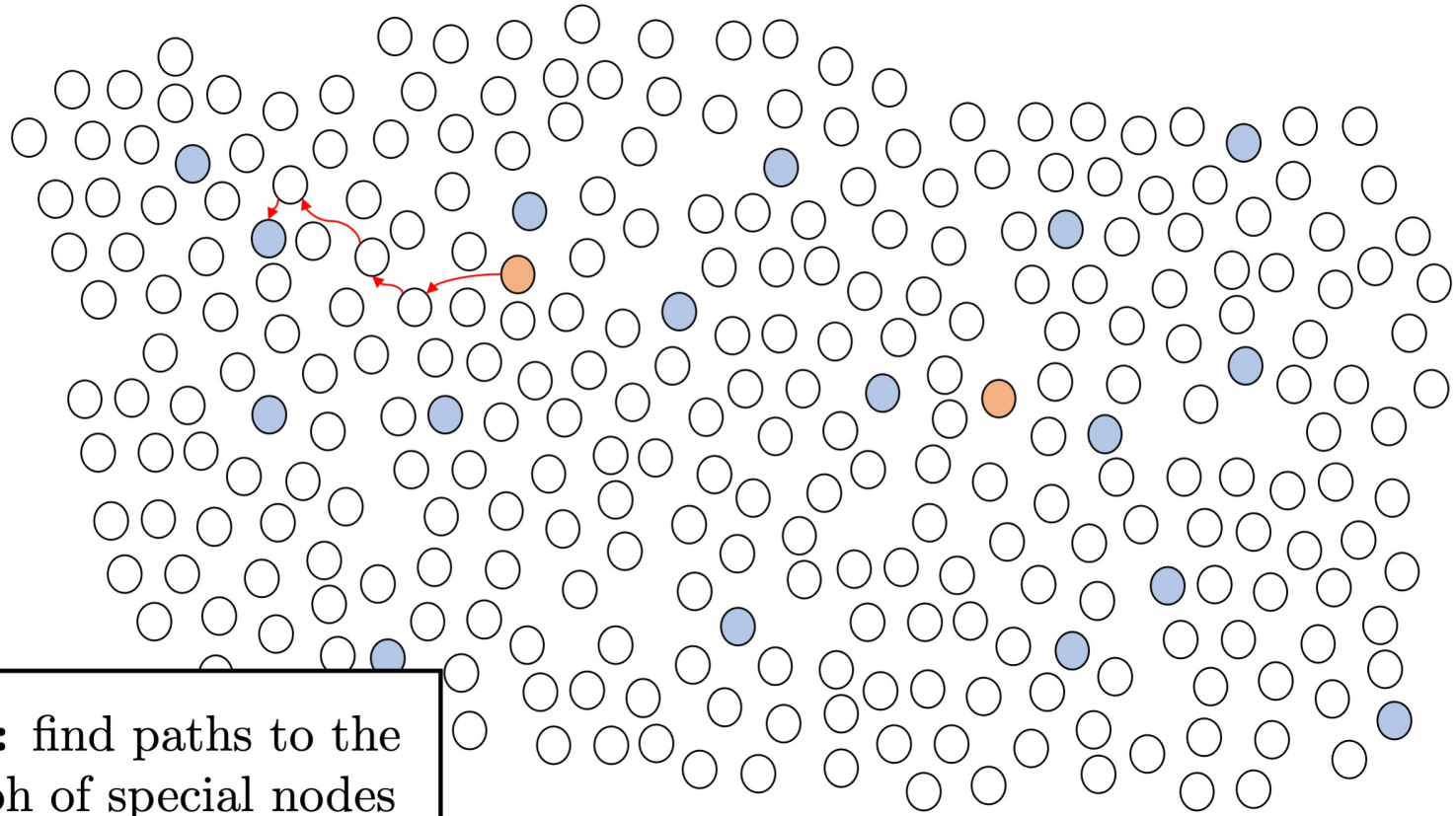
Step 1: find paths to the subgraph of special nodes

Reducing to Easier Problems using Special Subsets



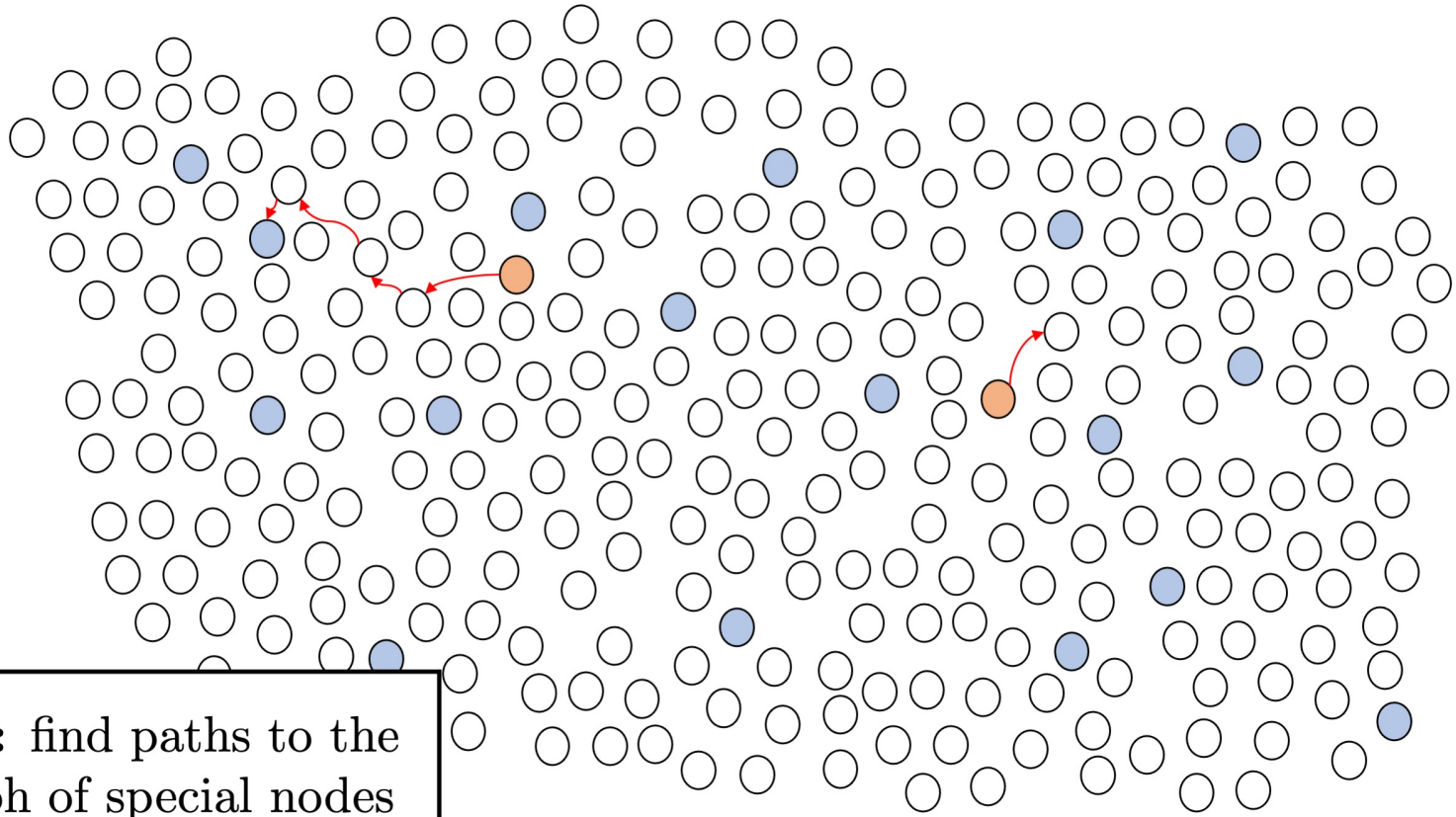
Step 1: find paths to the subgraph of special nodes

Reducing to Easier Problems using Special Subsets



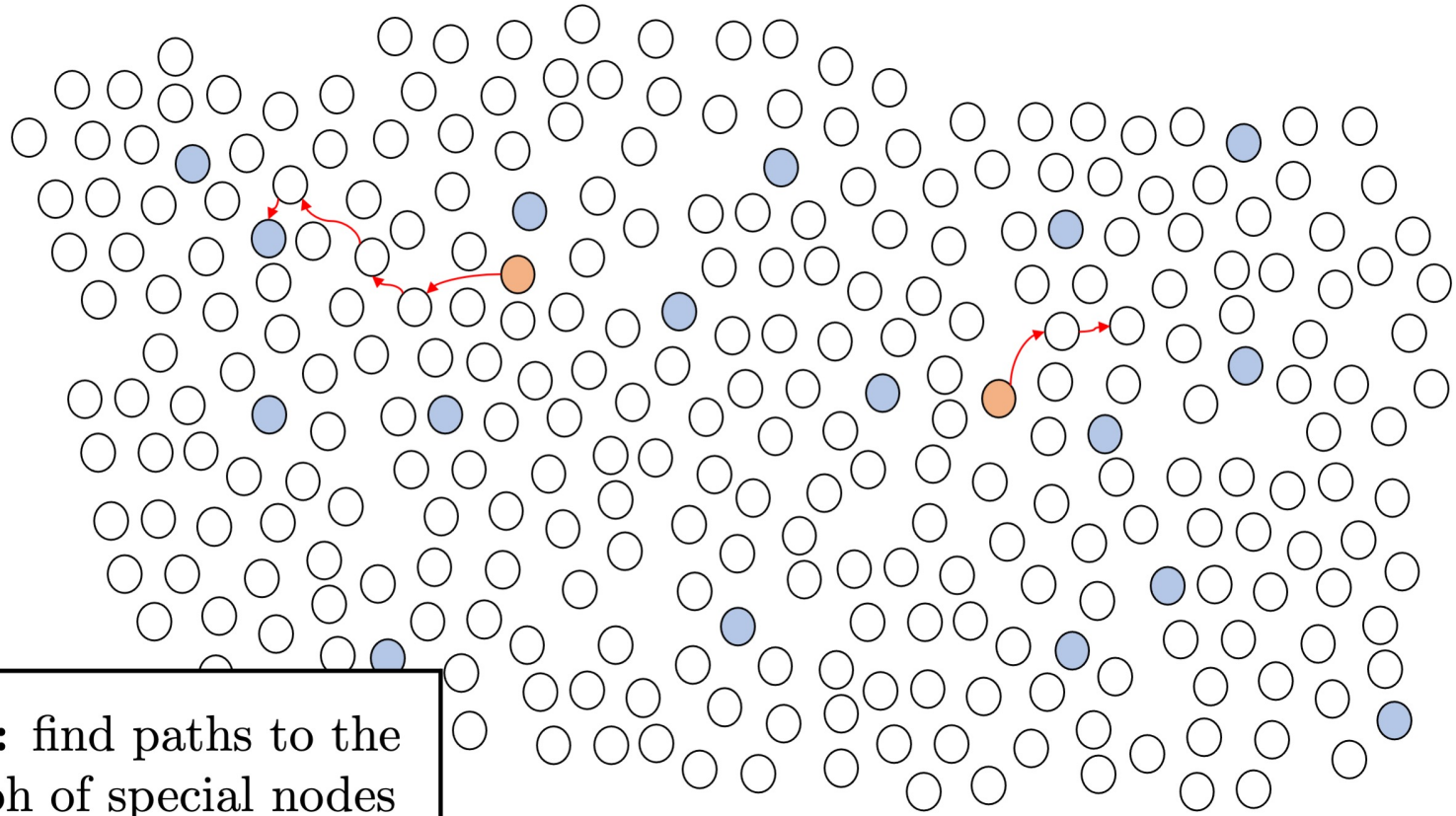
Step 1: find paths to the subgraph of special nodes

Reducing to Easier Problems using Special Subsets



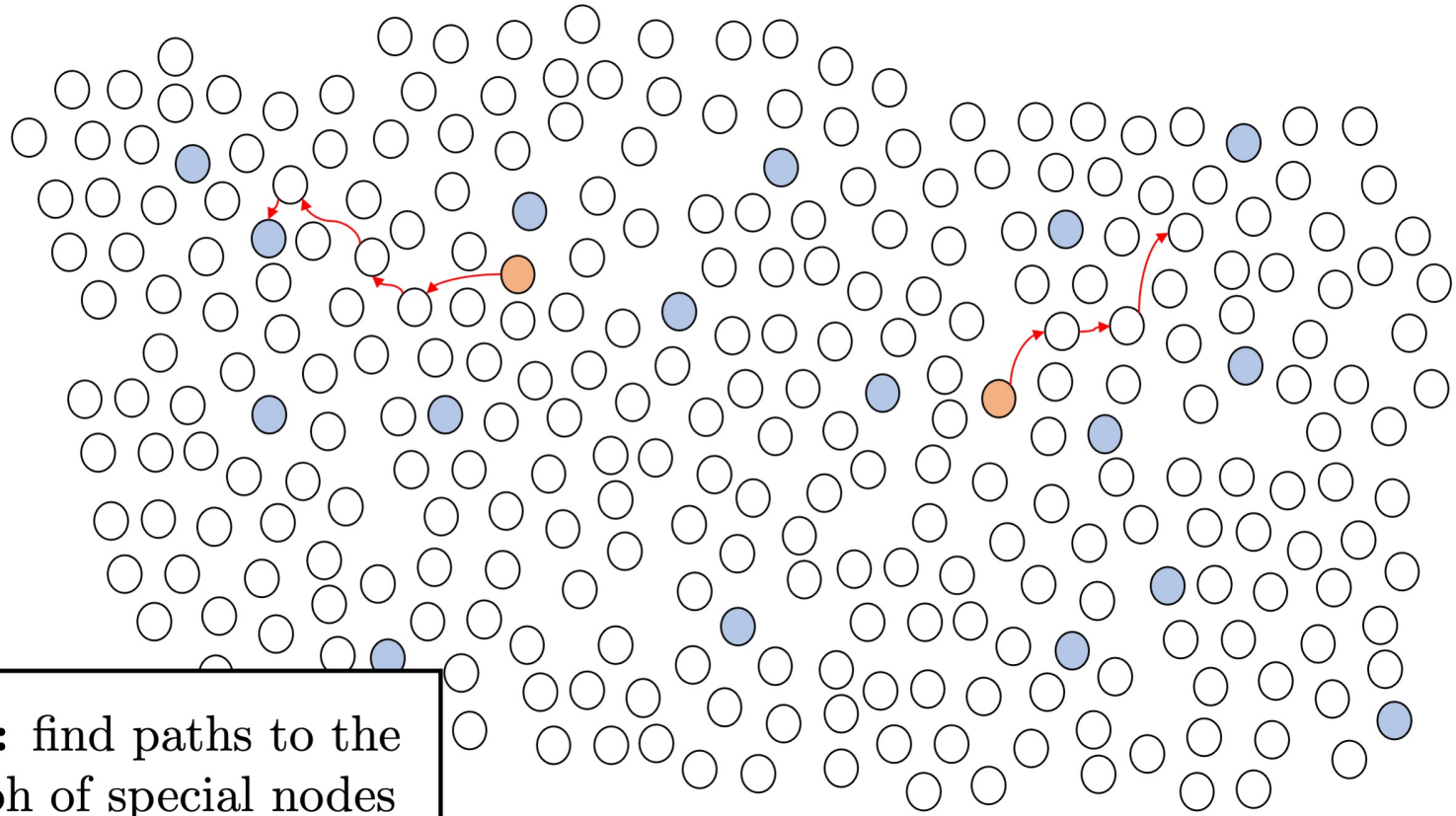
Step 1: find paths to the subgraph of special nodes

Reducing to Easier Problems using Special Subsets



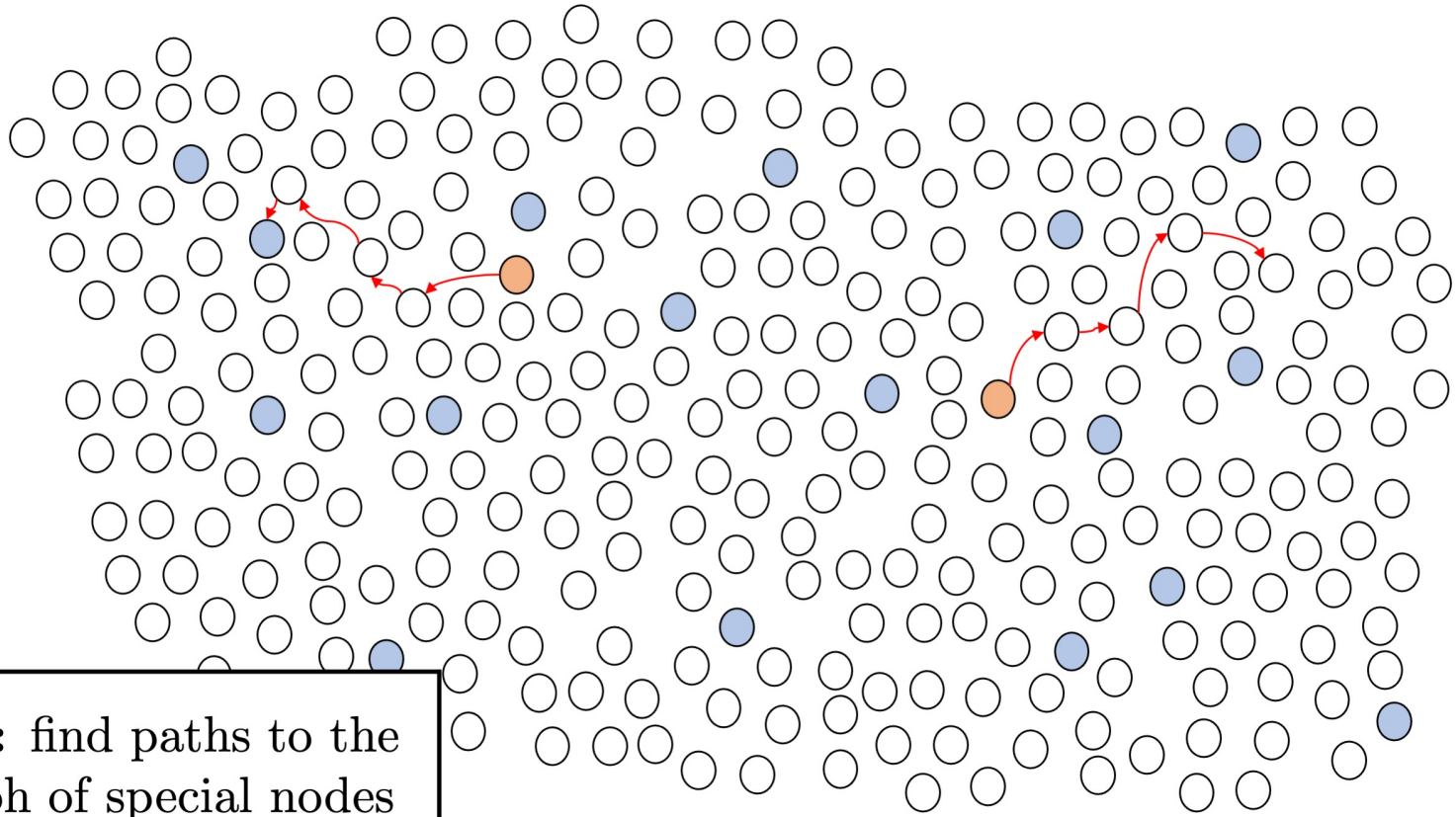
Step 1: find paths to the subgraph of special nodes

Reducing to Easier Problems using Special Subsets



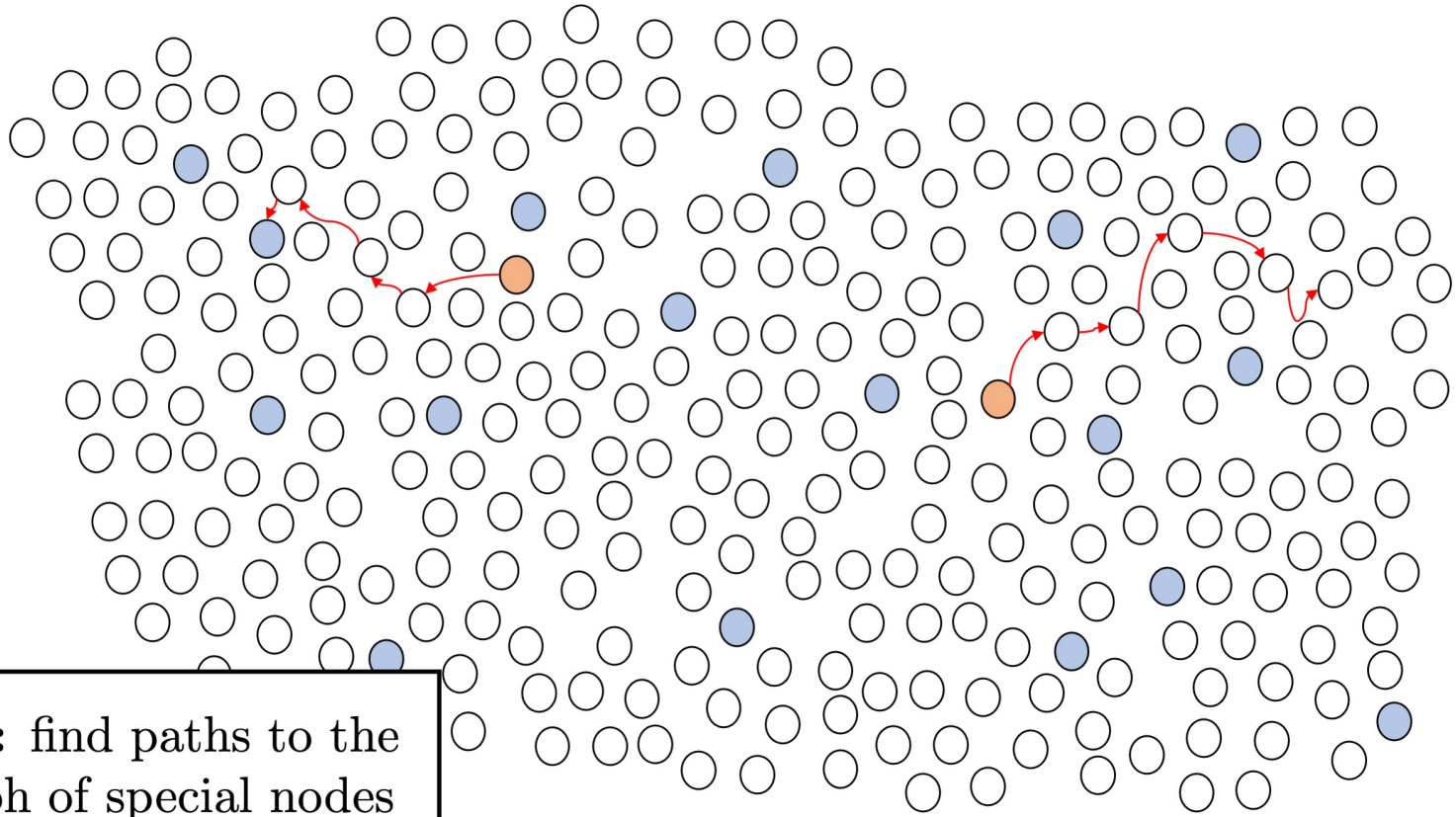
Step 1: find paths to the subgraph of special nodes

Reducing to Easier Problems using Special Subsets



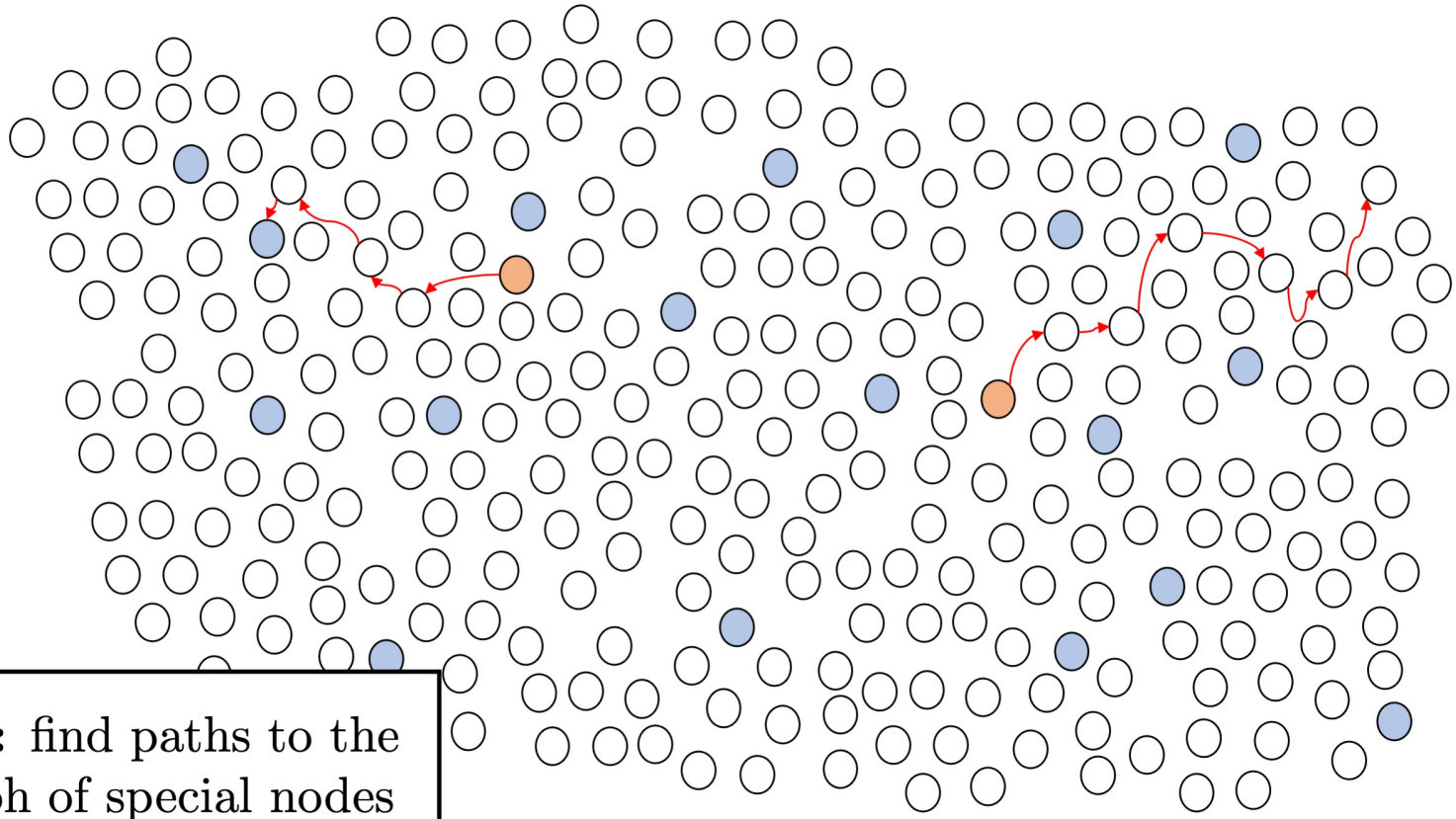
Step 1: find paths to the subgraph of special nodes

Reducing to Easier Problems using Special Subsets



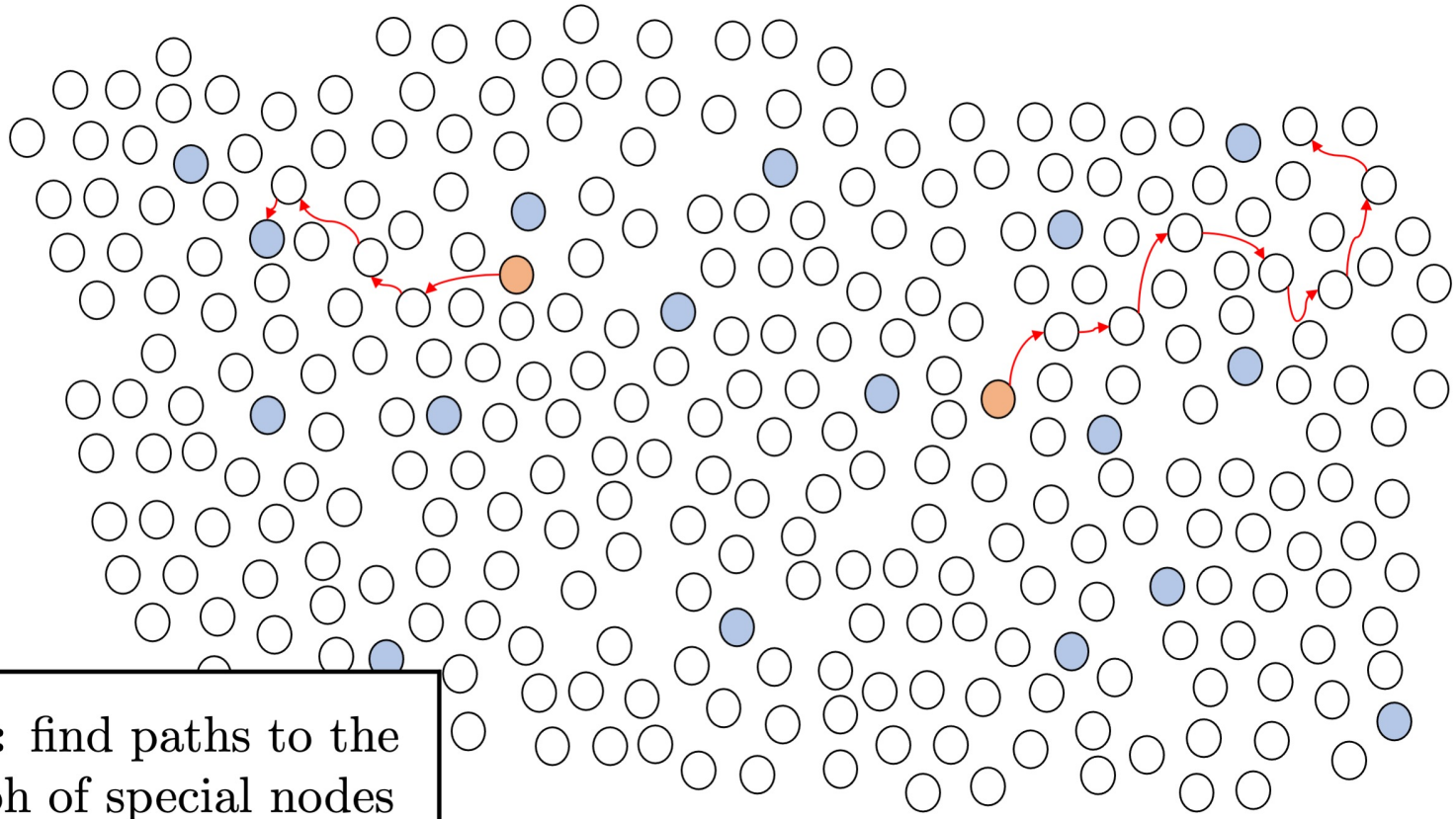
Step 1: find paths to the subgraph of special nodes

Reducing to Easier Problems using Special Subsets



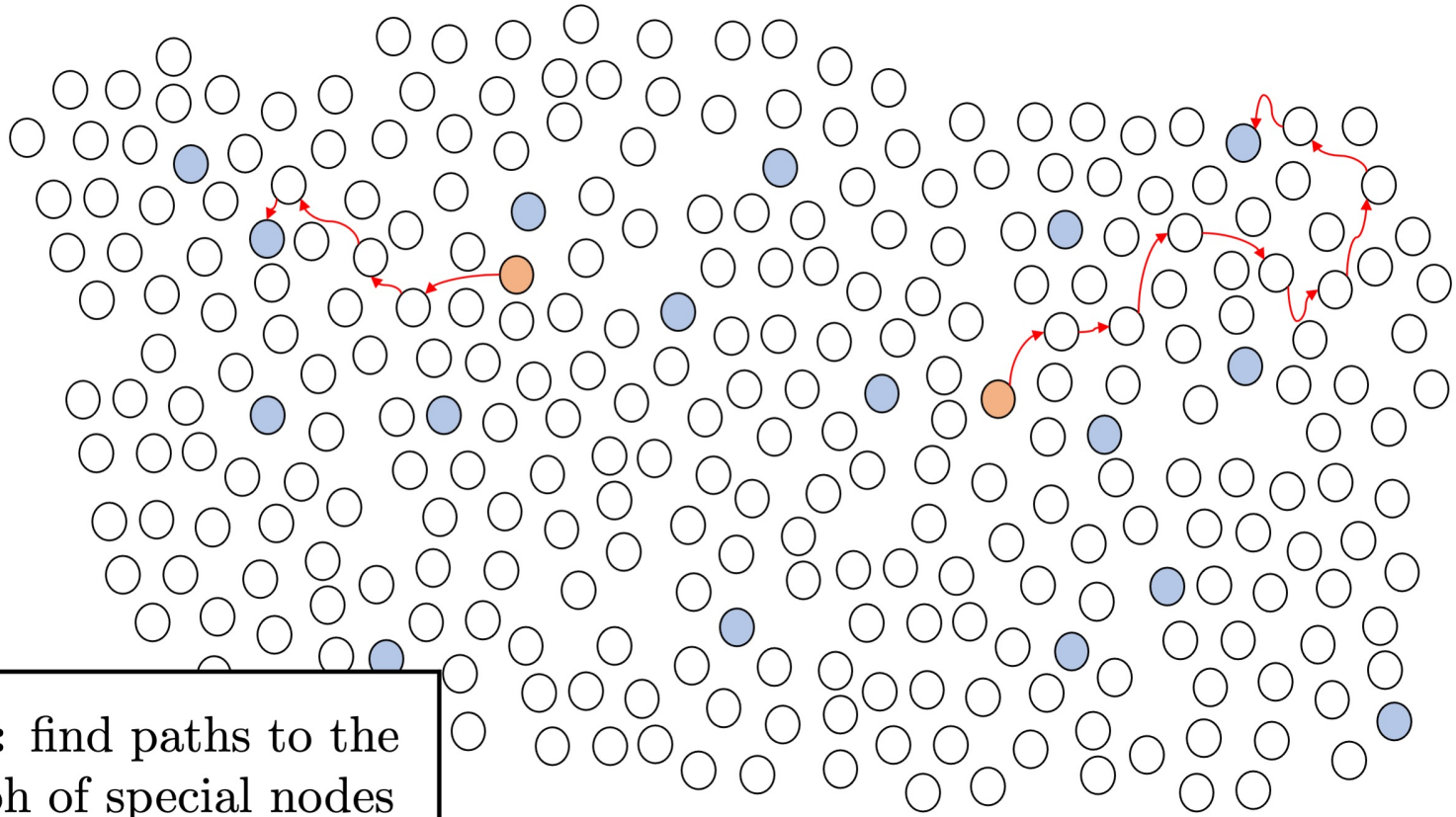
Step 1: find paths to the subgraph of special nodes

Reducing to Easier Problems using Special Subsets



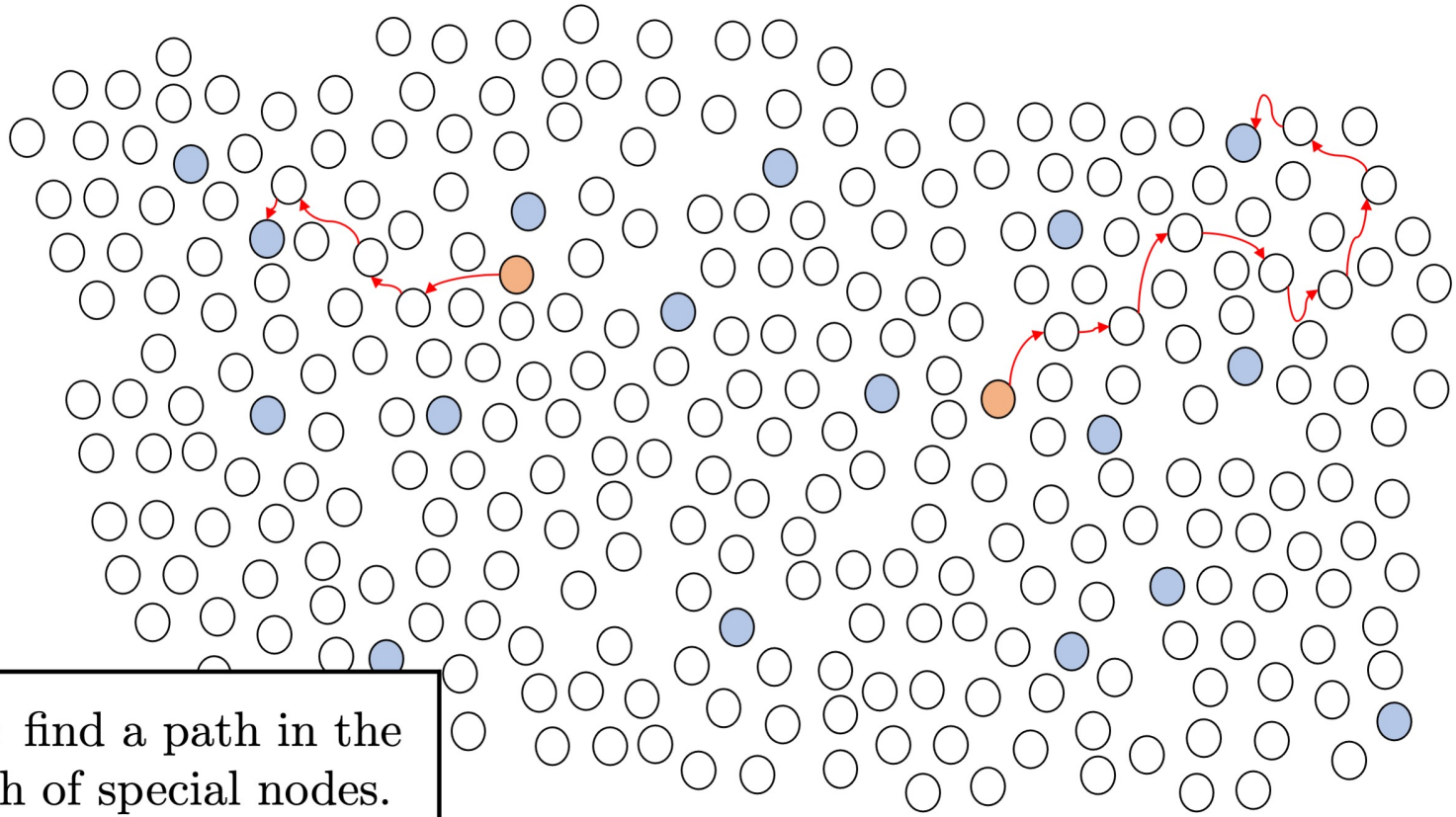
Step 1: find paths to the subgraph of special nodes

Reducing to Easier Problems using Special Subsets



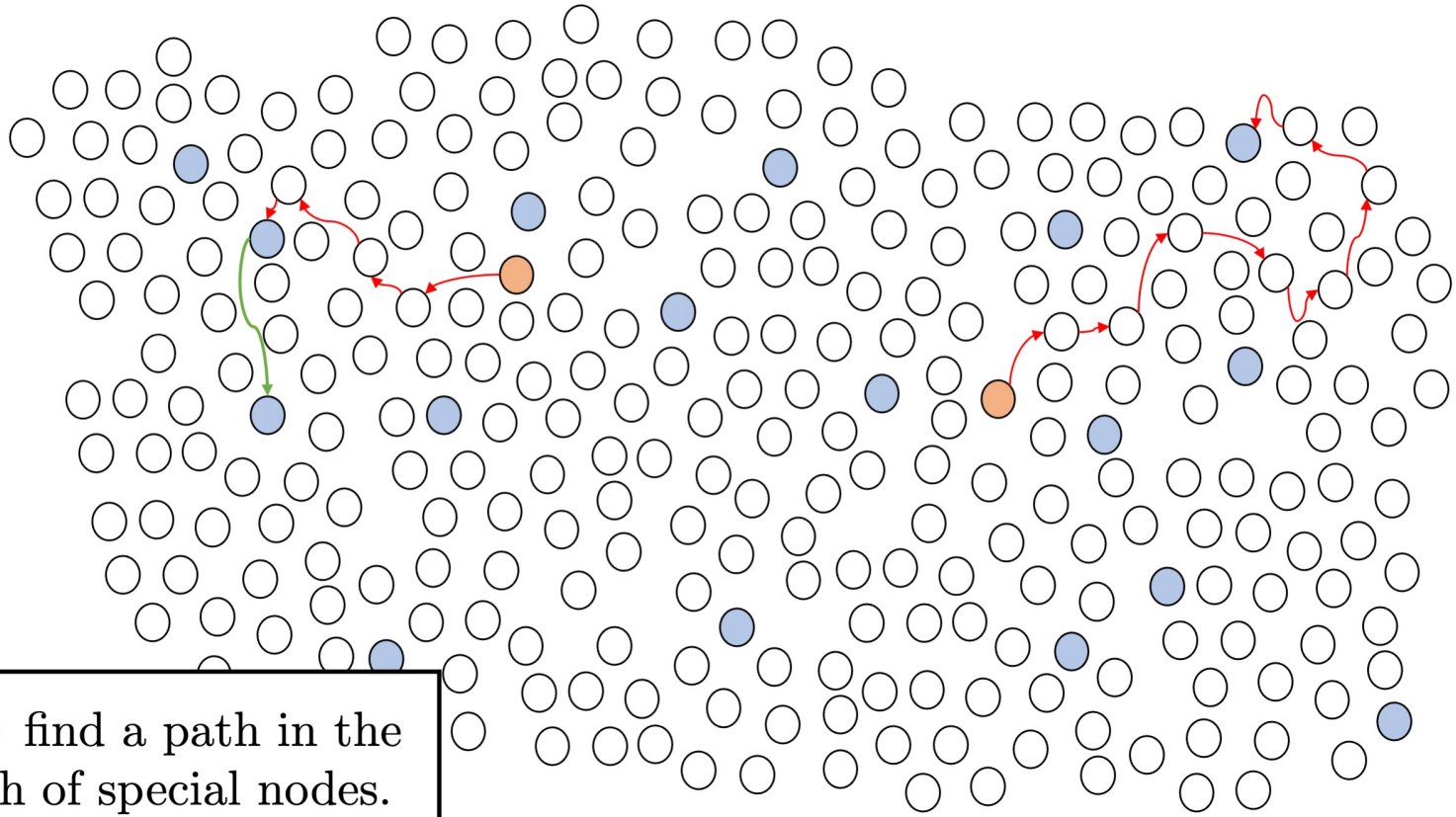
Step 1: find paths to the subgraph of special nodes

Reducing to Easier Problems using Special Subsets



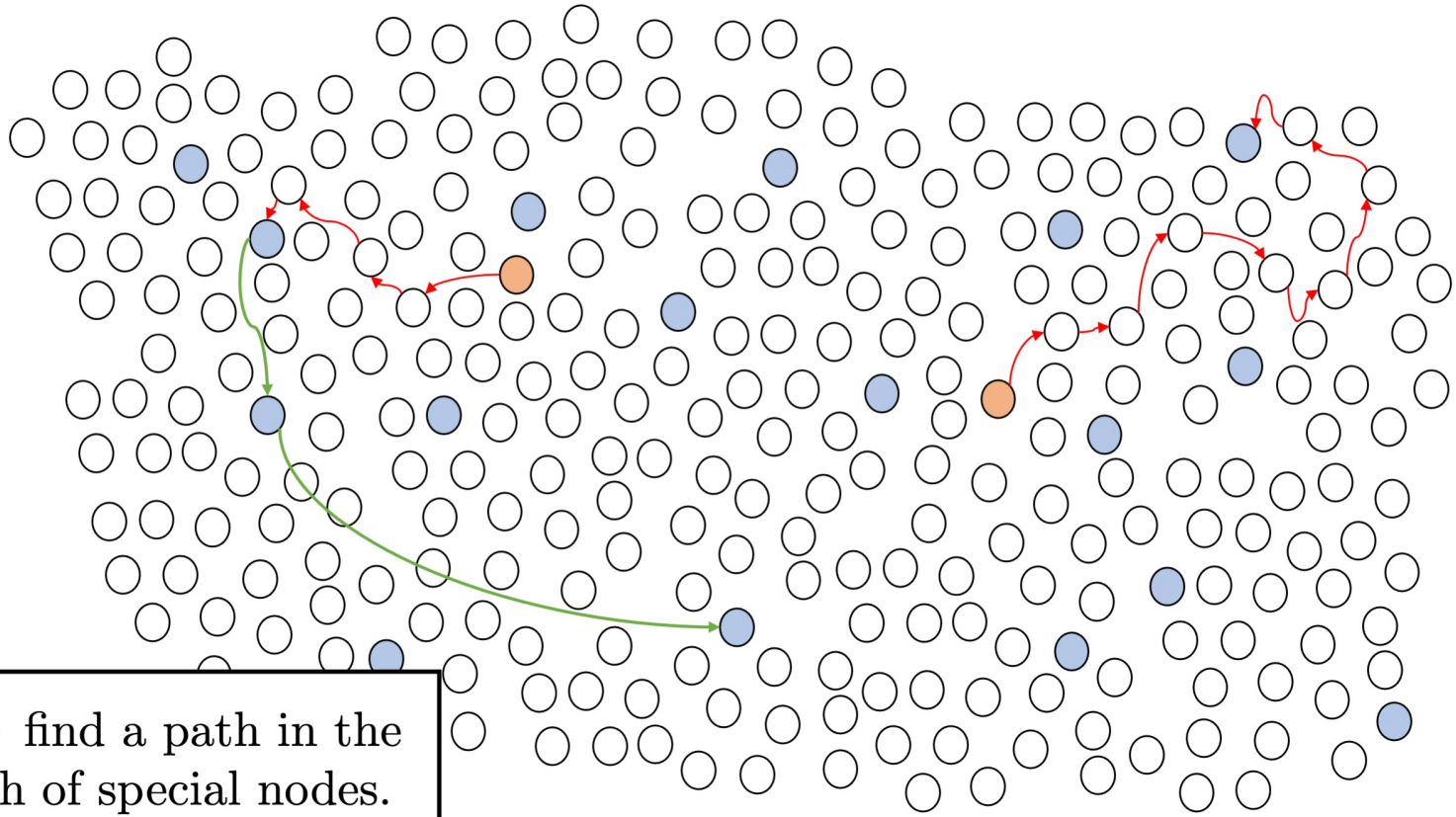
Step 2: find a path in the subgraph of special nodes.

Reducing to Easier Problems using Special Subsets



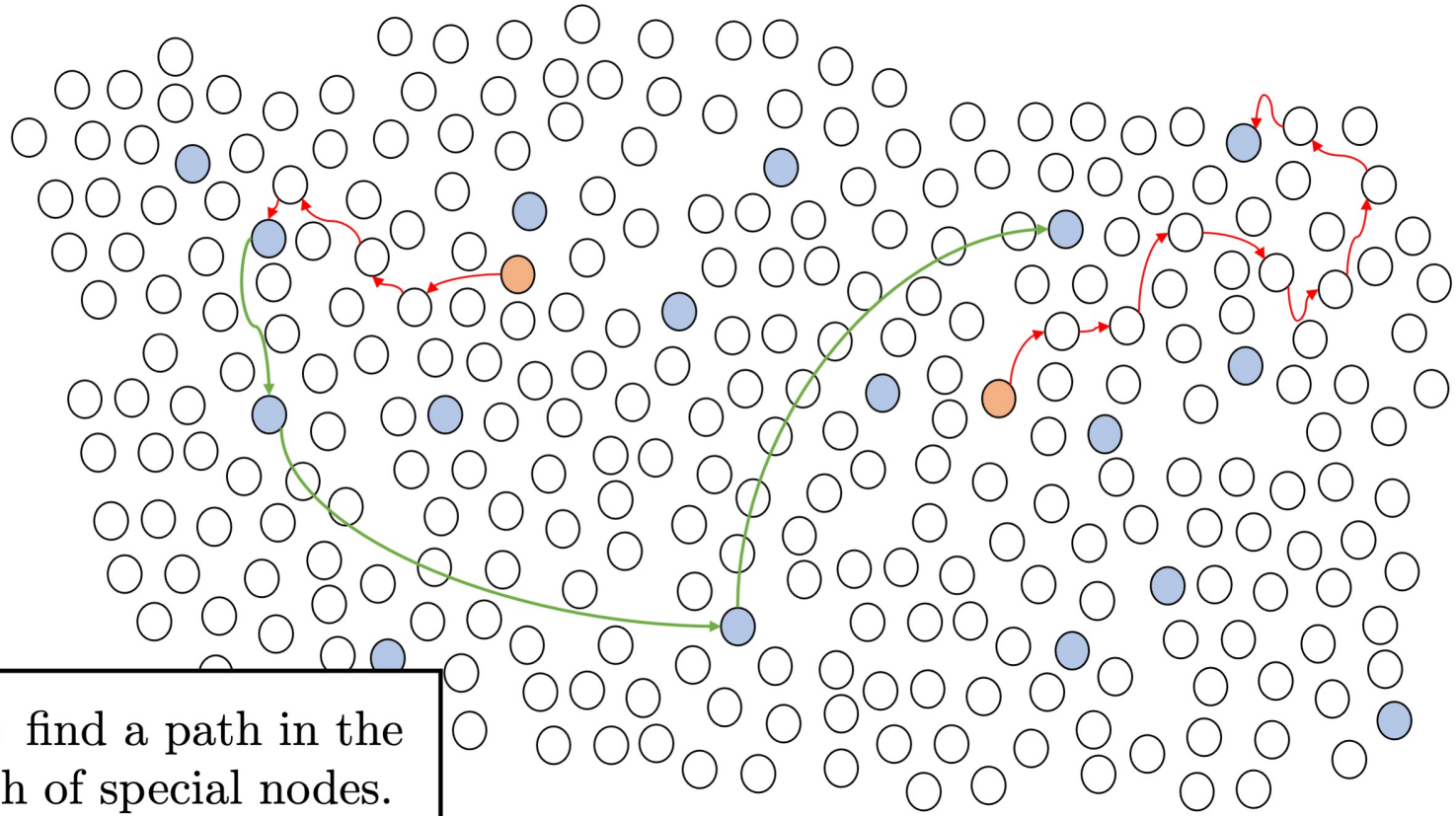
Step 2: find a path in the subgraph of special nodes.

Reducing to Easier Problems using Special Subsets



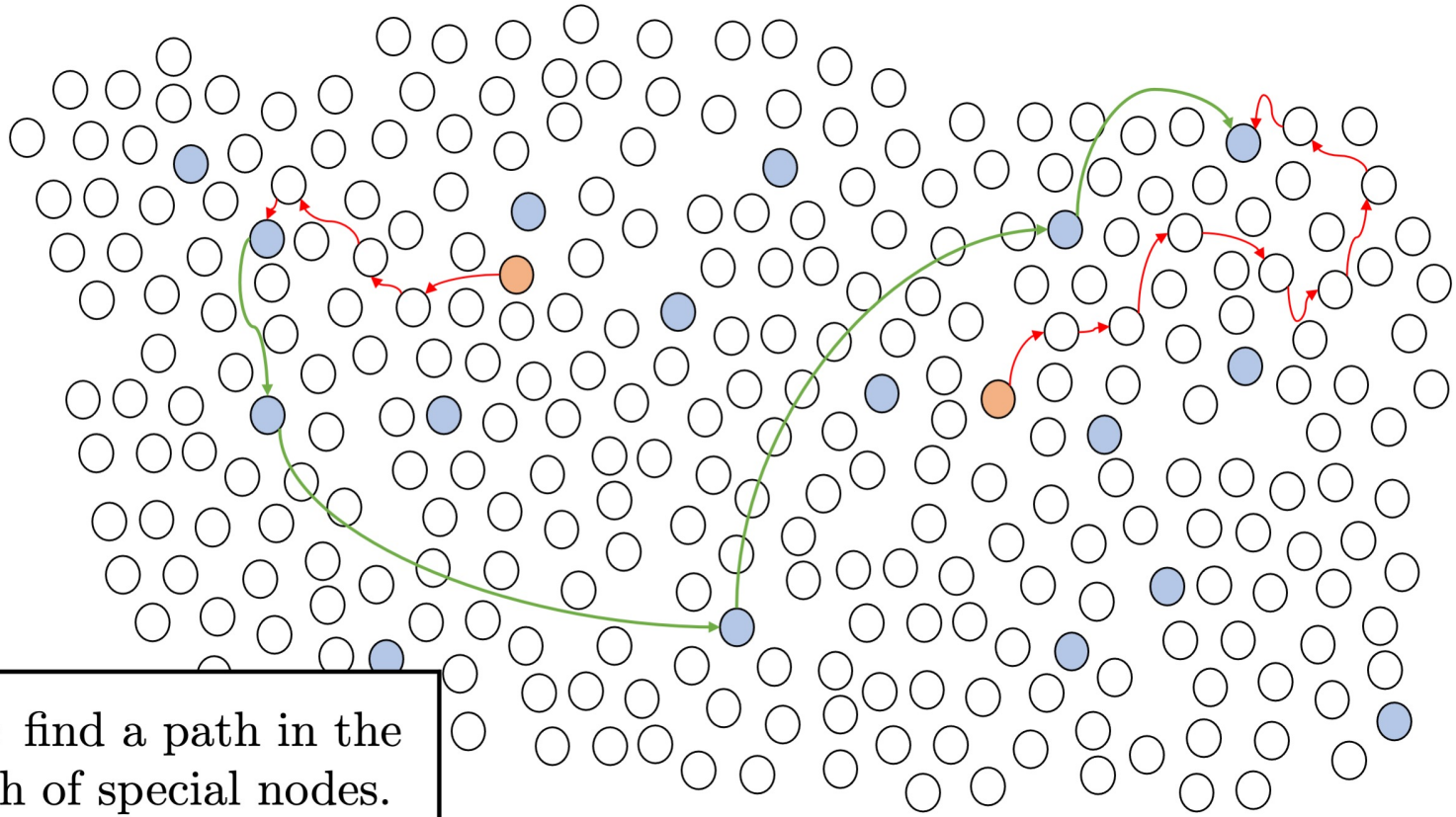
Step 2: find a path in the subgraph of special nodes.

Reducing to Easier Problems using Special Subsets



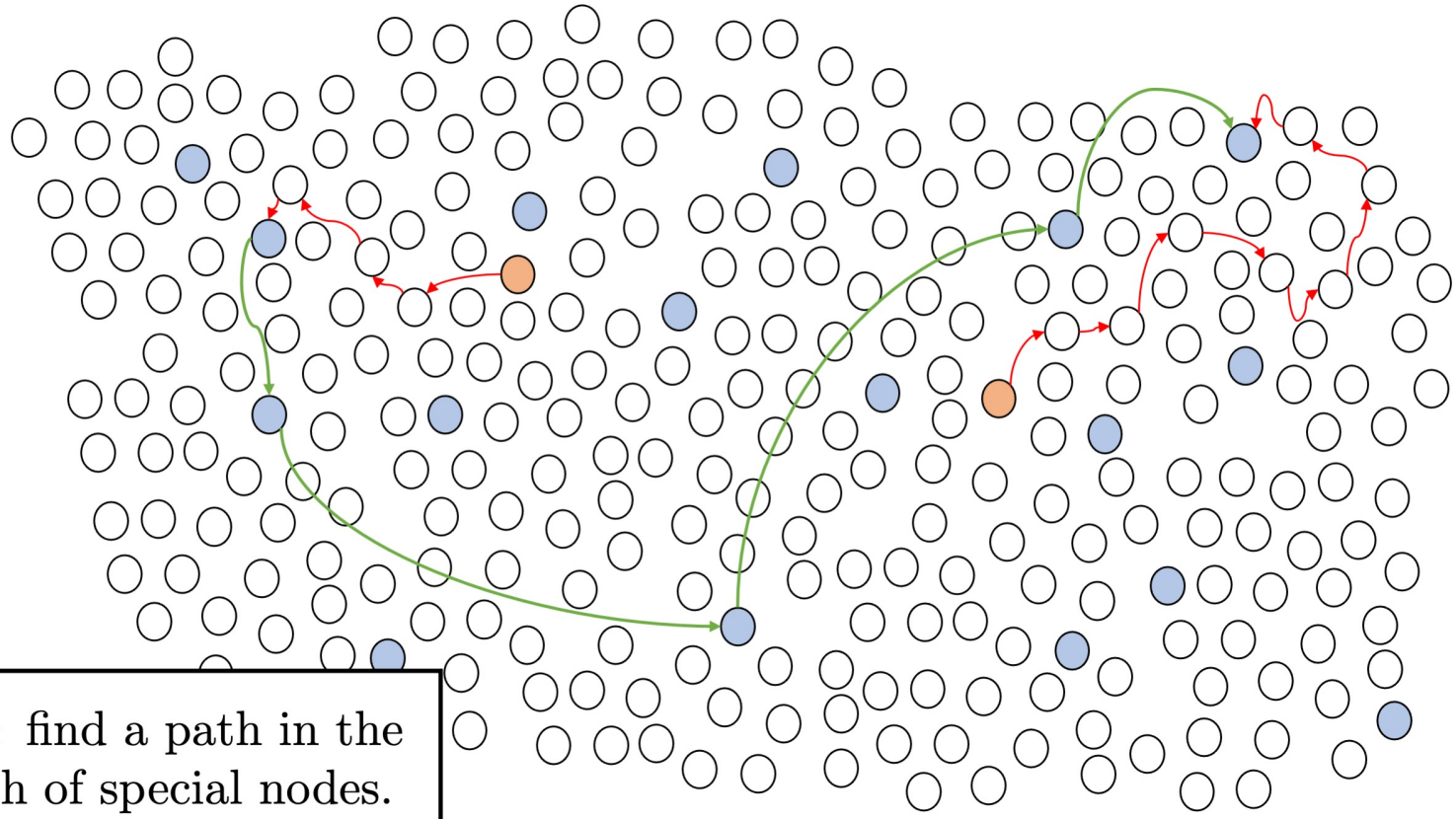
Step 2: find a path in the subgraph of special nodes.

Reducing to Easier Problems using Special Subsets



Step 2: find a path in the subgraph of special nodes.

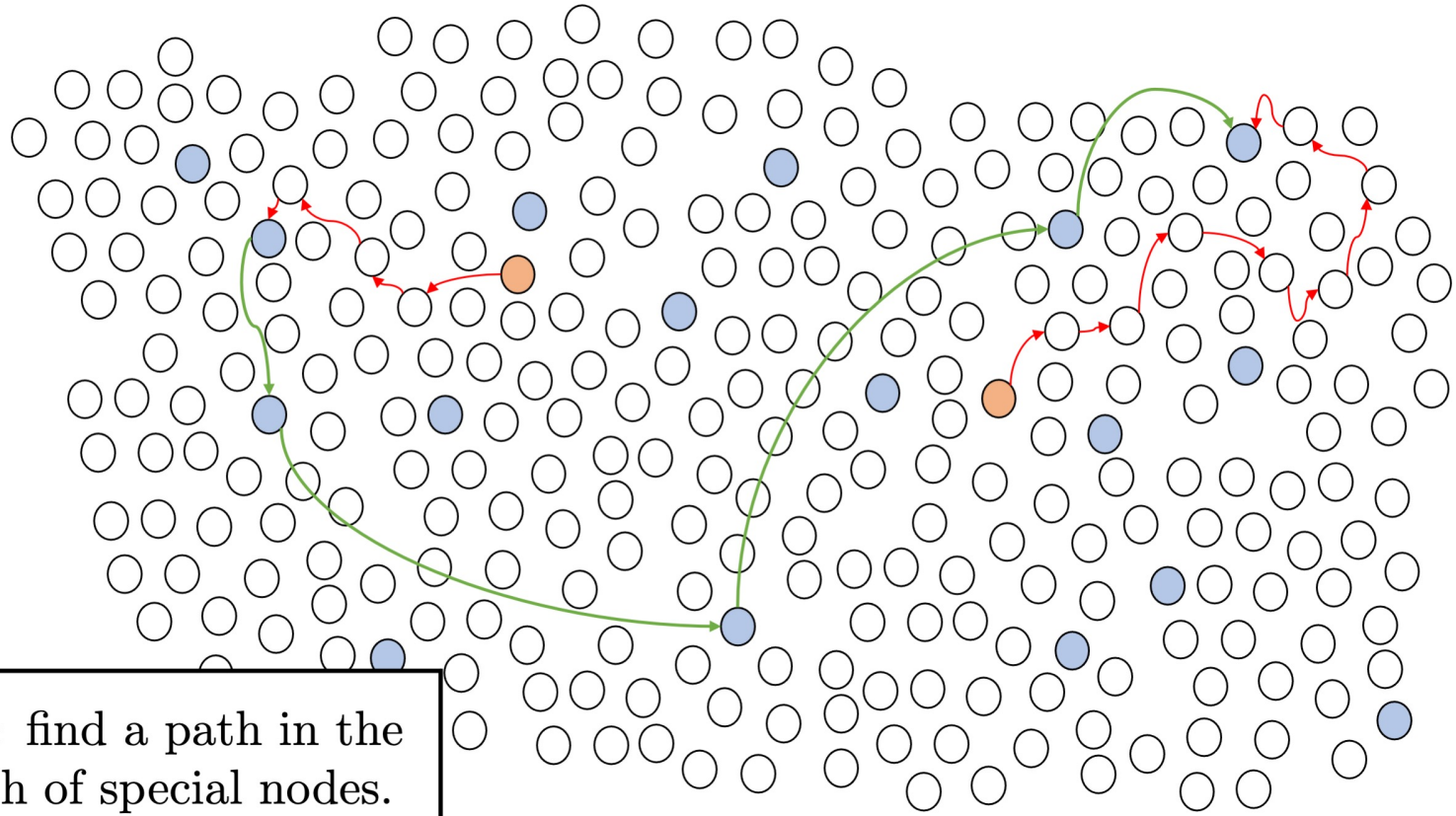
Reducing to Easier Problems using Special Subsets



Step 2: find a path in the subgraph of special nodes.

This should be *easier* so **Step 1** is bottleneck

Reducing to Easier Problems using Special Subsets



Step 2: find a path in the subgraph of special nodes.

Concatenate the paths to get the full path.

Improving Detection of the Special Subset

As Step 1 is the bottleneck, we focus on improving this part, i.e., walking in the graph until we reach a node in the special subgraph.

Improving Detection of the Special Subset

As Step 1 is the bottleneck, we focus on improving this part, i.e., walking in the graph until we reach a node in the special subgraph.

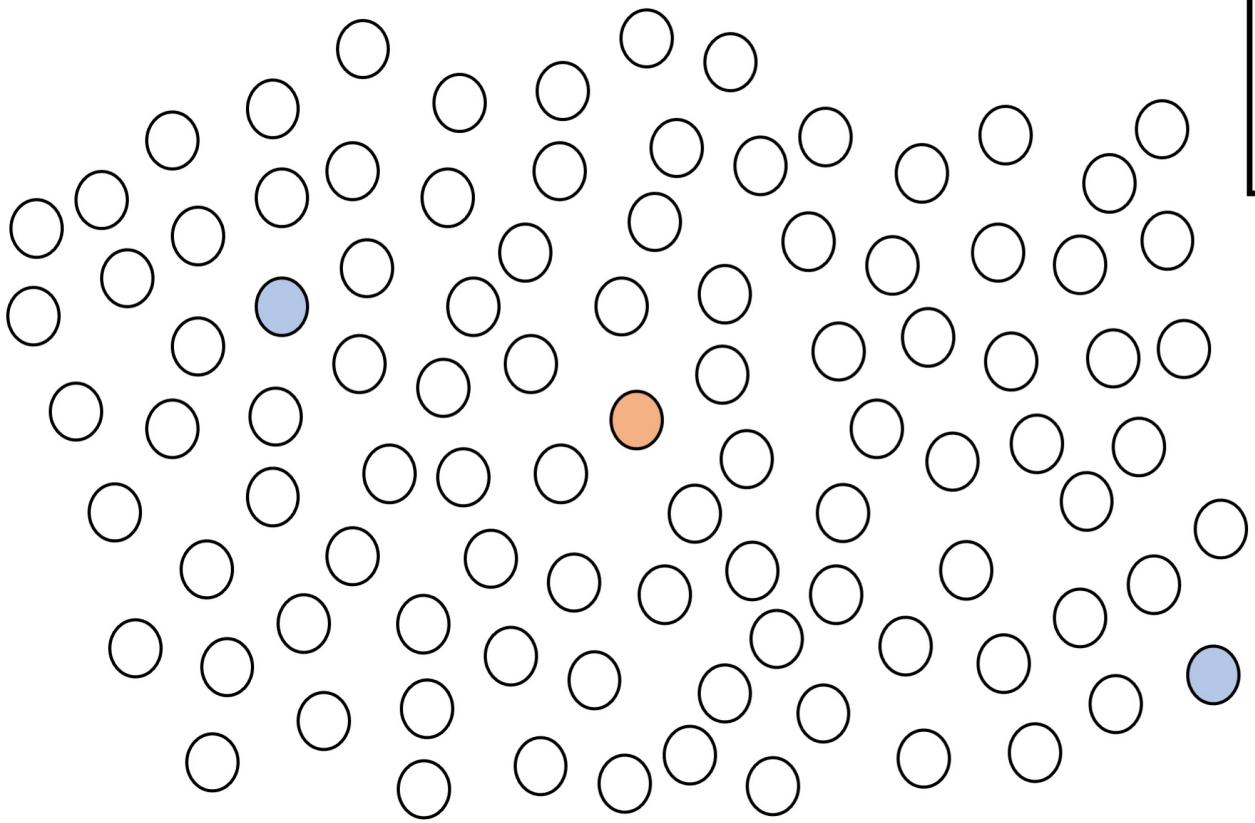
At each step of the walk, we reveal **one** node.

Improving Detection of the Special Subset

As Step 1 is the bottleneck, we focus on improving this part, i.e., walking in the graph until we reach a node in the special subgraph.

At each step of the walk, we reveal **one** node. Can we detect more nodes at each step in an efficient way?

Improving Detection of the Special Subset

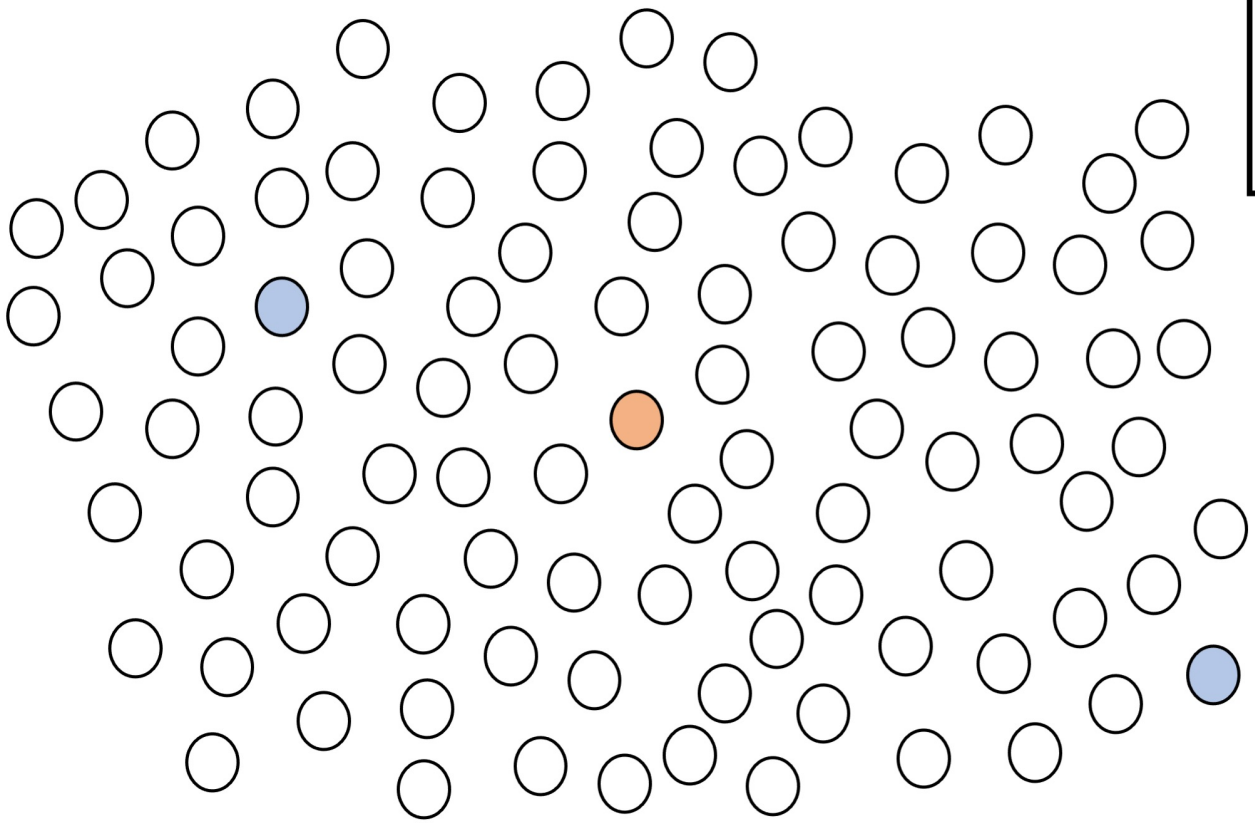


Step 1: Choose a detection set $\mathcal{D} = \{d_1, \ldots, d_n\}^*$

special nodes

*more on this later

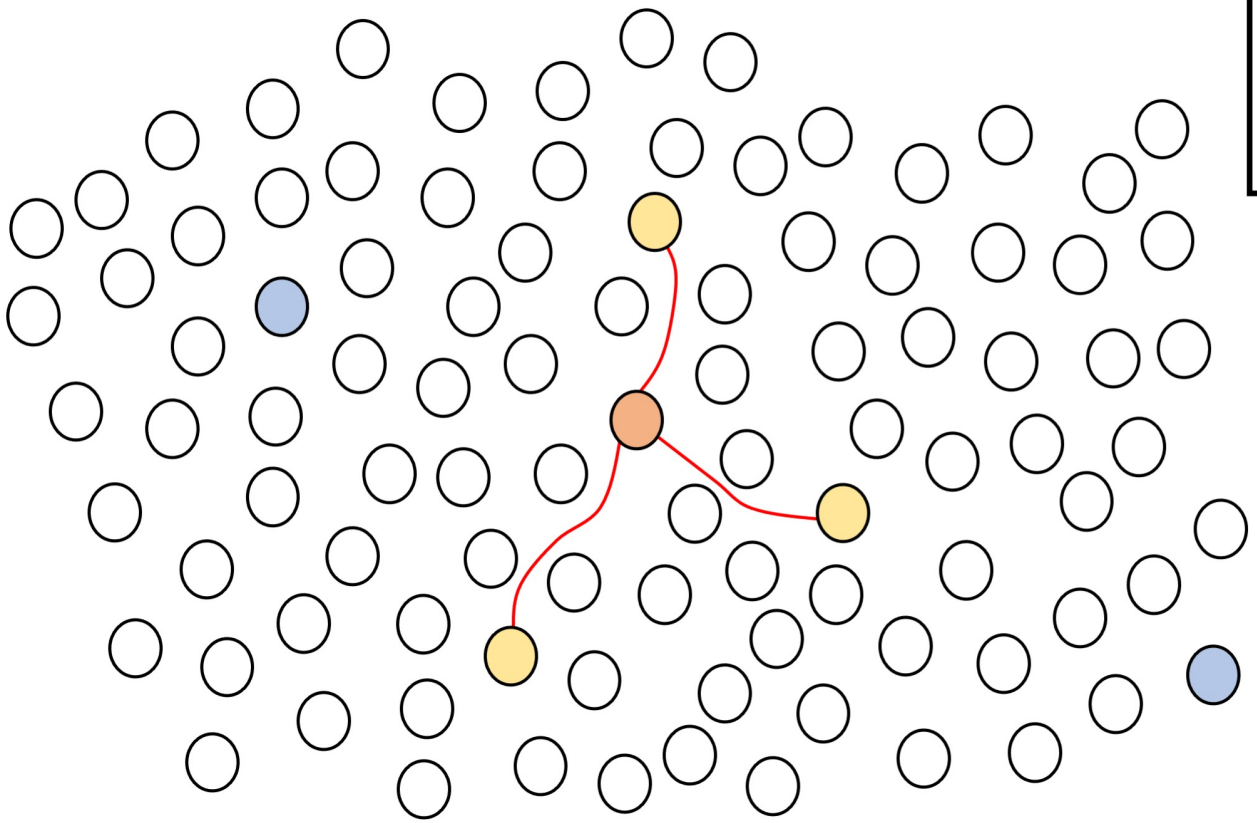
Improving Detection of the Special Subset



Step 2: Perform detection for d_1, $d_2, d_3 \ldots$

special nodes

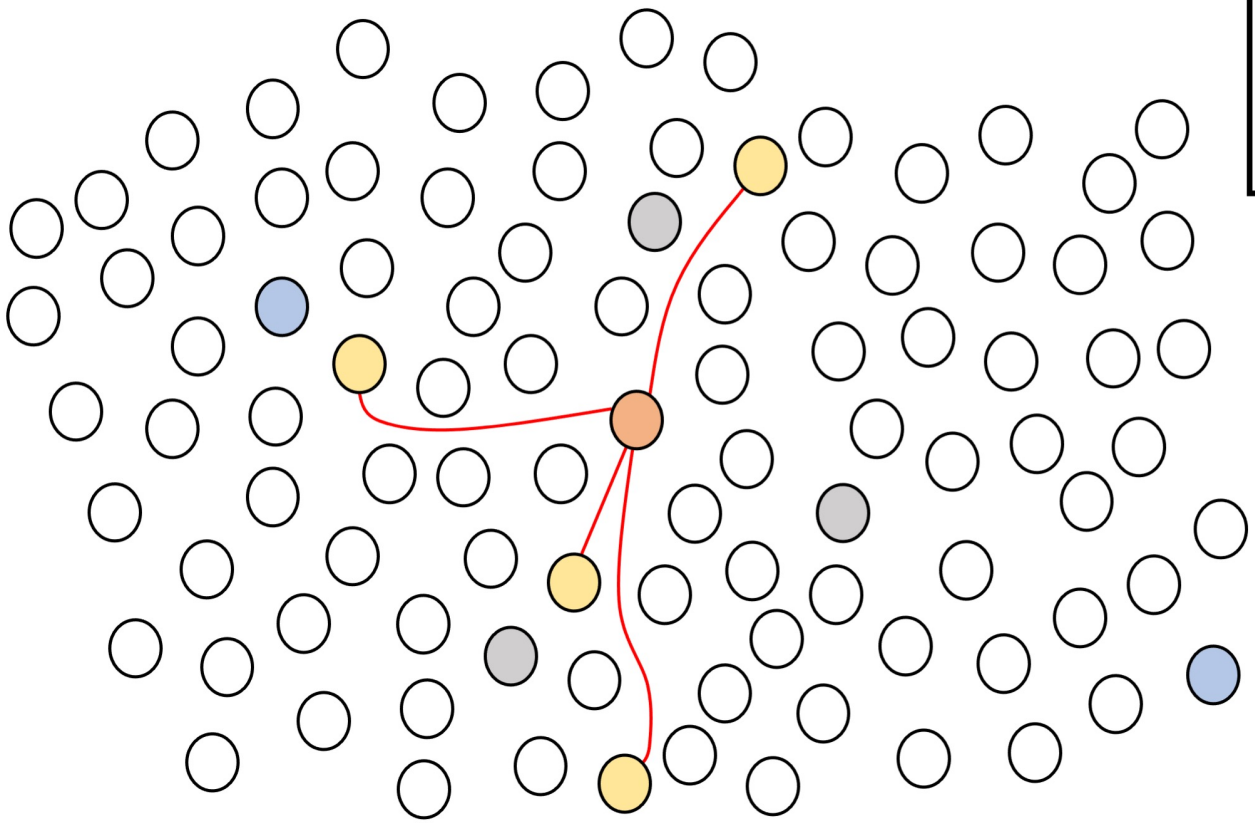
Improving Detection of the Special Subset



Step 2: Perform detection for d_1, $d_2, d_3 \ldots$

special nodes

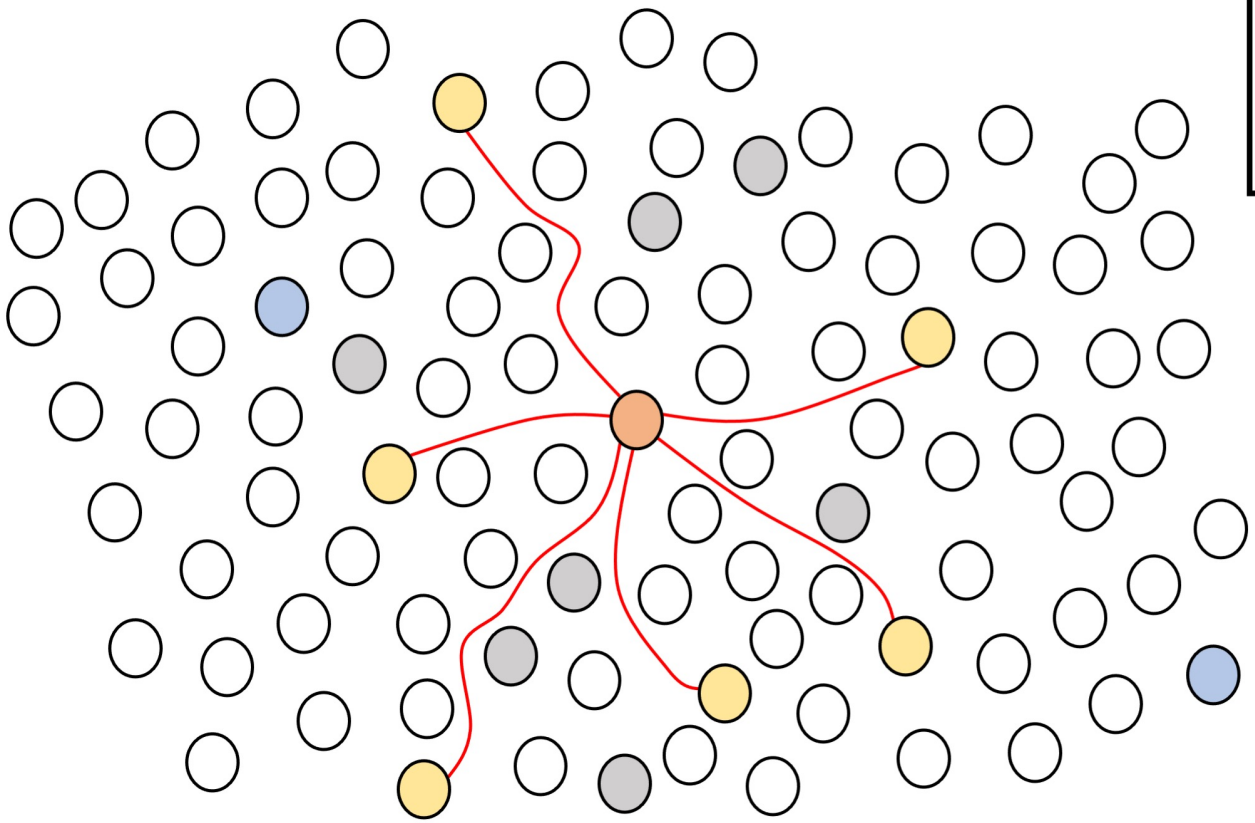
Improving Detection of the Special Subset



Step 2: Perform detection for d_1, $d_2, d_3 \ldots$

special nodes

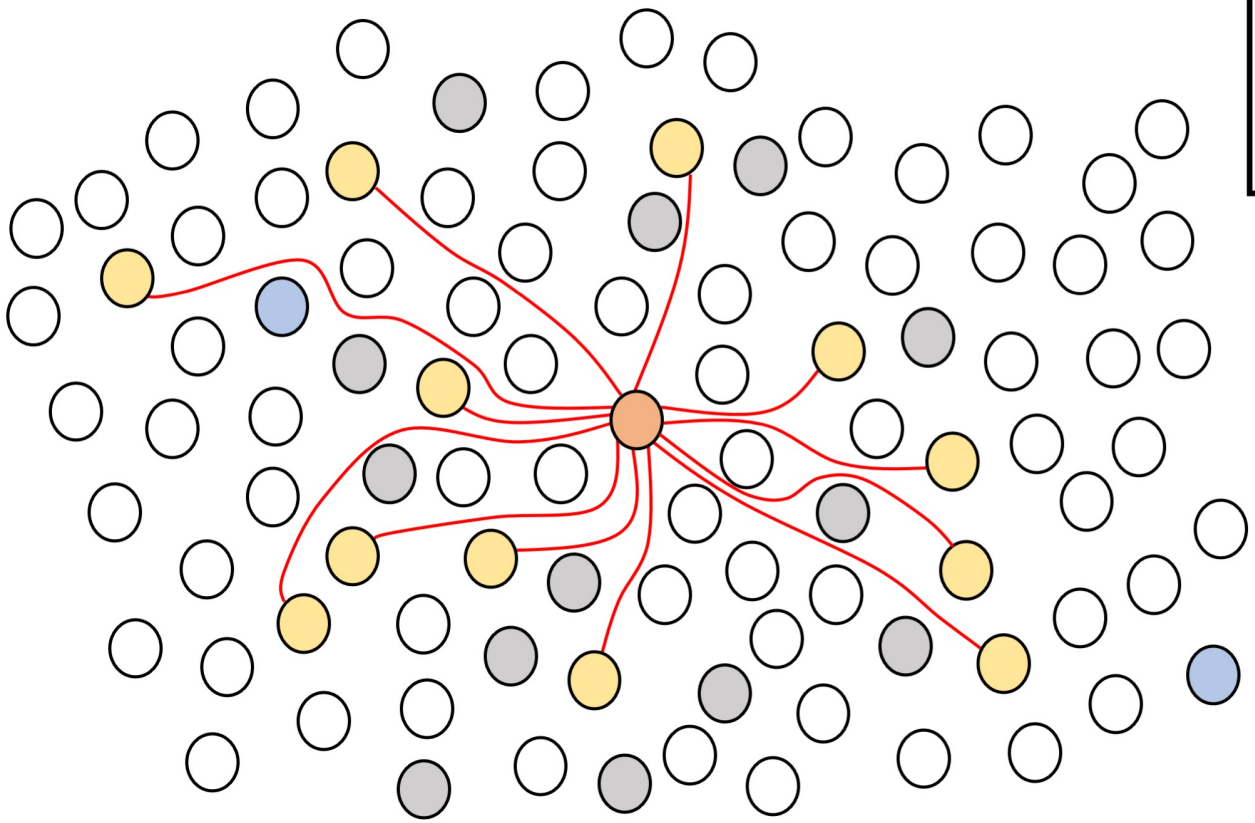
Improving Detection of the Special Subset



Step 2: Perform detection for d_1, $d_2, d_3 \ldots$

special nodes

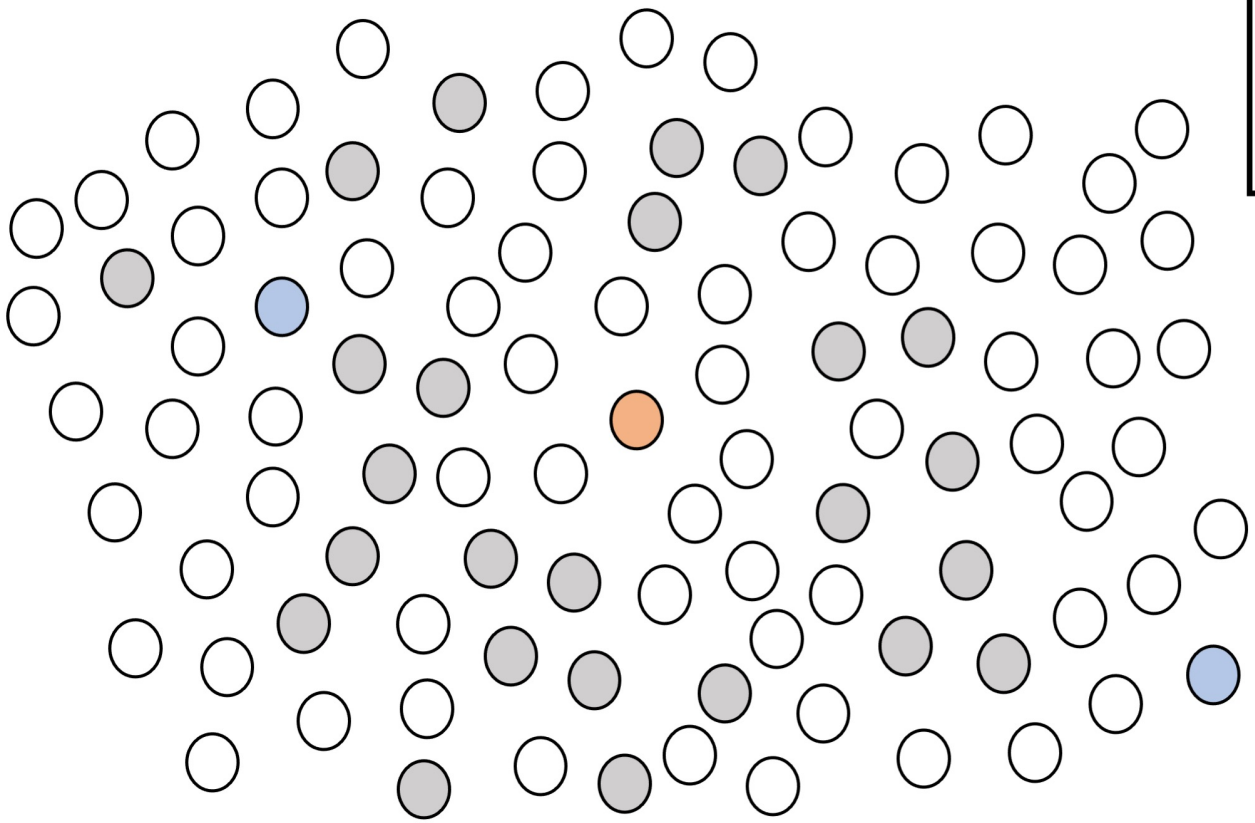
Improving Detection of the Special Subset



Step 2: Perform detection for d_1, $d_2, d_3 \ldots$

special nodes

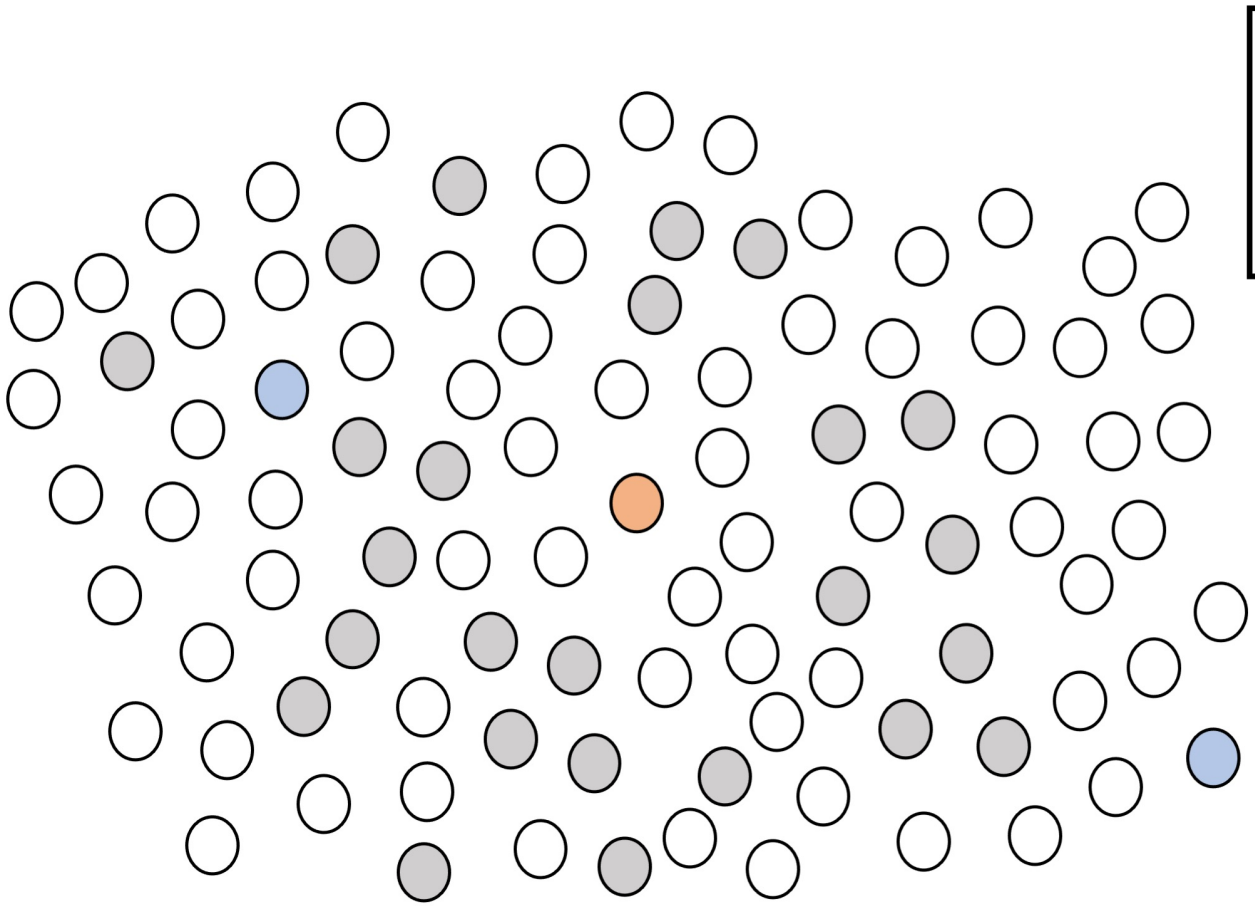
Improving Detection of the Special Subset



Step 2: Perform detection for d_1, $d_2, d_3 \ldots$

special nodes

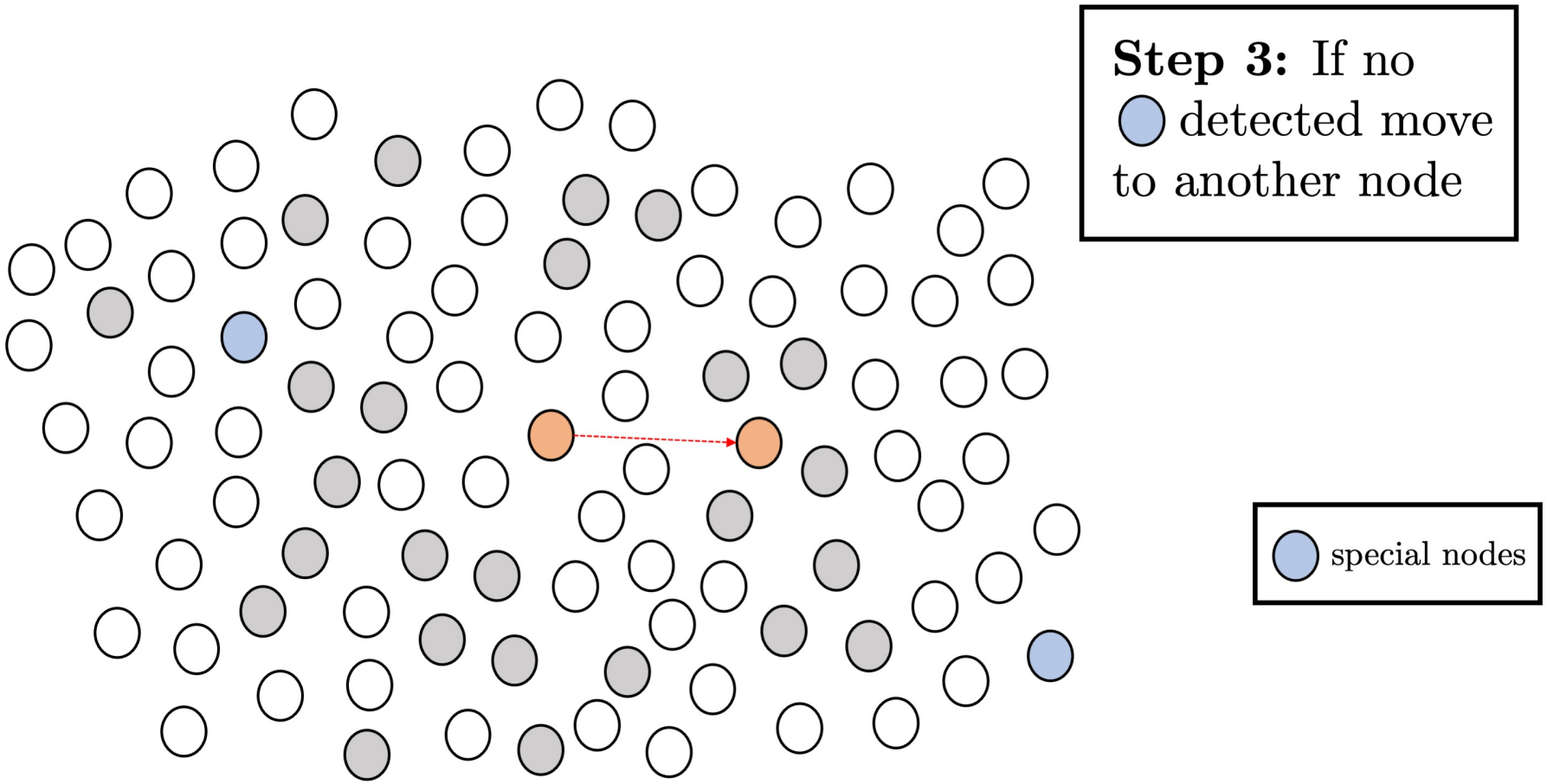
Improving Detection of the Special Subset



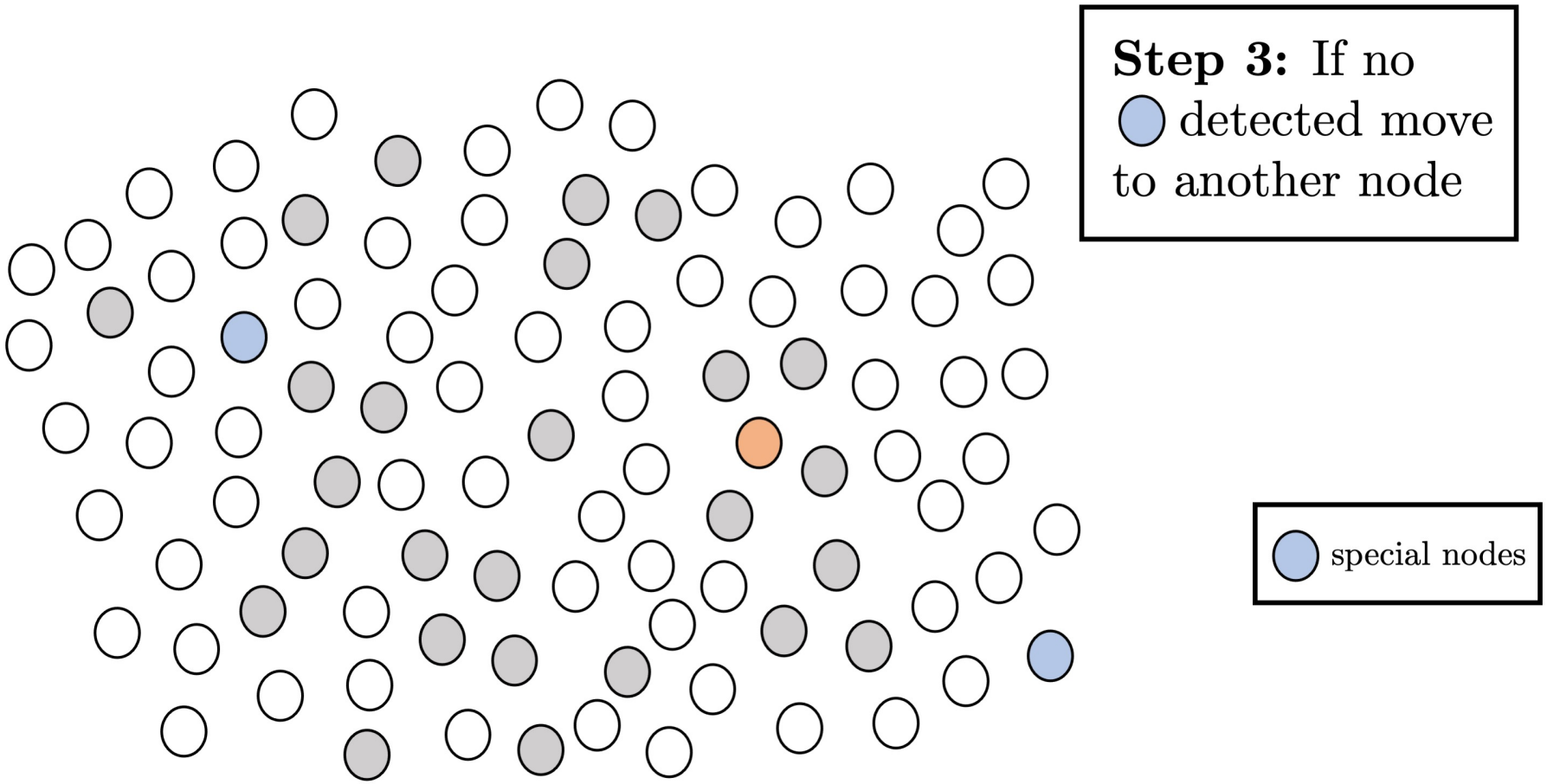
Step 3: If no detected move to another node

special nodes

Improving Detection of the Special Subset



Improving Detection of the Special Subset



Choosing Detection Set

Suppose we can perform detection for elements in some $\mathcal{S}$.

Choosing Detection Set

Suppose we can perform detection for elements in some $\mathcal{S}$. We want to choose a set $\mathcal{D} \subset \mathcal{S}$ on which to perform our detection.

Choosing Detection Set

Suppose we can perform detection for elements in some $\mathcal{S}$. We want to choose a set $\mathcal{D} \subset \mathcal{S}$ on which to perform our detection.

How do we choose $\mathcal{D}$?

Choosing Detection Set

Suppose we can perform detection for elements in some $\mathcal{S}$. We want to choose a set $\mathcal{D} \subset \mathcal{S}$ on which to perform our detection.

How do we choose $\mathcal{D}$? Performing detection for elements in $\mathcal{S}$ adds extra cost to each step in the graph, but more nodes are revealed.

Choosing Detection Set

Suppose we can perform detection for elements in some $\mathcal{S}$. We want to choose a set $\mathcal{D} \subset \mathcal{S}$ on which to perform our detection.

How do we choose $\mathcal{D}$? Performing detection for elements in $\mathcal{S}$ adds extra cost to each step in the graph, but more nodes are revealed.

We want to choose $\mathcal{D} \subset \mathcal{S}$ that minimises

$$\frac{\text{cost}}{\text{nodes revealed}},$$

where nodes revealed $=$ nodes walked on *and* detected.

Choosing Detection Set

Suppose we can perform detection for elements in some $\mathcal{S}$. We want to choose a set $\mathcal{D} \subset \mathcal{S}$ on which to perform our detection.

How do we choose $\mathcal{D}$? Performing detection for elements in $\mathcal{S}$ adds extra cost to each step in the graph, but more nodes are revealed.

We want to choose $\mathcal{D} \subset \mathcal{S}$ that minimises

$$\frac{\text{cost}}{\text{nodes revealed}},$$

where nodes revealed $=$ nodes walked on *and* detected.

If detection cost is *small* compared to nodes revealed, then we could hope to improve the concrete complexity of the attack.

Application to Dimension One

Based on joint work with Craig Costello, Jia Shi (eprint 2021/1488): SuperSolver

How can we apply these ideas to solving the general isogeny problem in dimension 1: given supersingular, $E_1, E_2/\mathbb{F}_{p^2}$ find an isogeny $\phi : E_1 \to E_2$?

Application to Dimension One

Based on joint work with Craig Costello, Jia Shi (eprint 2021/1488): SuperSolver

How can we apply these ideas to solving the general isogeny problem in dimension 1: given supersingular, $E_1, E_2/\mathbb{F}_{p^2}$ find an isogeny $\phi : E_1 \to E_2$?

Recall that we can reframe this as a path finding problem in the supersingular isogeny graph: find a path between nodes j_1, j_2.

Application to Dimension One

Based on joint work with Craig Costello, Jia Shi (eprint 2021/1488): SuperSolver

How can we apply these ideas to solving the general isogeny problem in dimension 1: given supersingular, $E_1, E_2/\mathbb{F}_{p^2}$ find an isogeny $\phi : E_1 \to E_2$?

Recall that we can reframe this as a path finding problem in the supersingular isogeny graph: find a path between nodes j_1, j_2.

First, how do we take steps in this graph?

Application to Dimension One

The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One: how to take steps

Application to Dimension One

The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One: how to take steps

We use *modular polynomials*.

Application to Dimension One

The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One: how to take steps

We use *modular polynomials*. The modular polynomial (of level N) $\Phi_N(X, Y) \in \mathbb{Z}[X, Y]$[2] parameterizes pairs of N-isogenous elliptic curves in terms of their j-invariants.

[2] Reducing coefficients mod p we can work with $\Phi_{N,p}(X, Y) \in \mathbb{F}_p[X, Y]$.

Application to Dimension One

The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One: how to take steps

We use *modular polynomials*. The modular polynomial (of level N) $\Phi_N(X, Y) \in \mathbb{Z}[X, Y]$[2] parameterizes pairs of N-isogenous elliptic curves in terms of their j-invariants.
It is symmetric and of degree D_N in both X and Y, where

$$D_N := \prod_{i=1}^{n} (\ell_i + 1)\ell_i^{e_i - 1}, \text{ for prime decomposition } \prod_{i=1}^{n} \ell_i^{e_i} \text{ of } N.$$

$D_N = N + 1$ for N prime.

[2] Reducing coefficients mod p we can work with $\Phi_{N,p}(X, Y) \in \mathbb{F}_p[X, Y]$.

Application to Dimension One

The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One: how to take steps

We use *modular polynomials*. The modular polynomial (of level N) $\Phi_N(X, Y) \in \mathbb{Z}[X, Y]$[2] parameterizes pairs of N-isogenous elliptic curves in terms of their j-invariants.
It is symmetric and of degree D_N in both X and Y, where

$$D_N := \prod_{i=1}^{n} (\ell_i + 1)\ell_i^{e_i - 1}, \text{ for prime decomposition } \prod_{i=1}^{n} \ell_i^{e_i} \text{ of } N.$$

$D_N = N + 1$ for N prime.

$\Phi_N(j_1, j_2) = 0 \iff j_1, j_2$ are j-invariants of N-isogenous elliptic curves.

[2] Reducing coefficients mod p we can work with $\Phi_{N,p}(X, Y) \in \mathbb{F}_p[X, Y]$.

Application to Dimension One

The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One: how to take steps

We use *modular polynomials*. The modular polynomial (of level N) $\Phi_N(X, Y) \in \mathbb{Z}[X, Y]$[2] parameterizes pairs of N-isogenous elliptic curves in terms of their j-invariants.
It is symmetric and of degree D_N in both X and Y, where

$$D_N := \prod_{i=1}^{n} (\ell_i + 1)\ell_i^{e_i - 1}, \text{ for prime decomposition } \prod_{i=1}^{n} \ell_i^{e_i} \text{ of } N.$$

$D_N = N + 1$ for N prime.

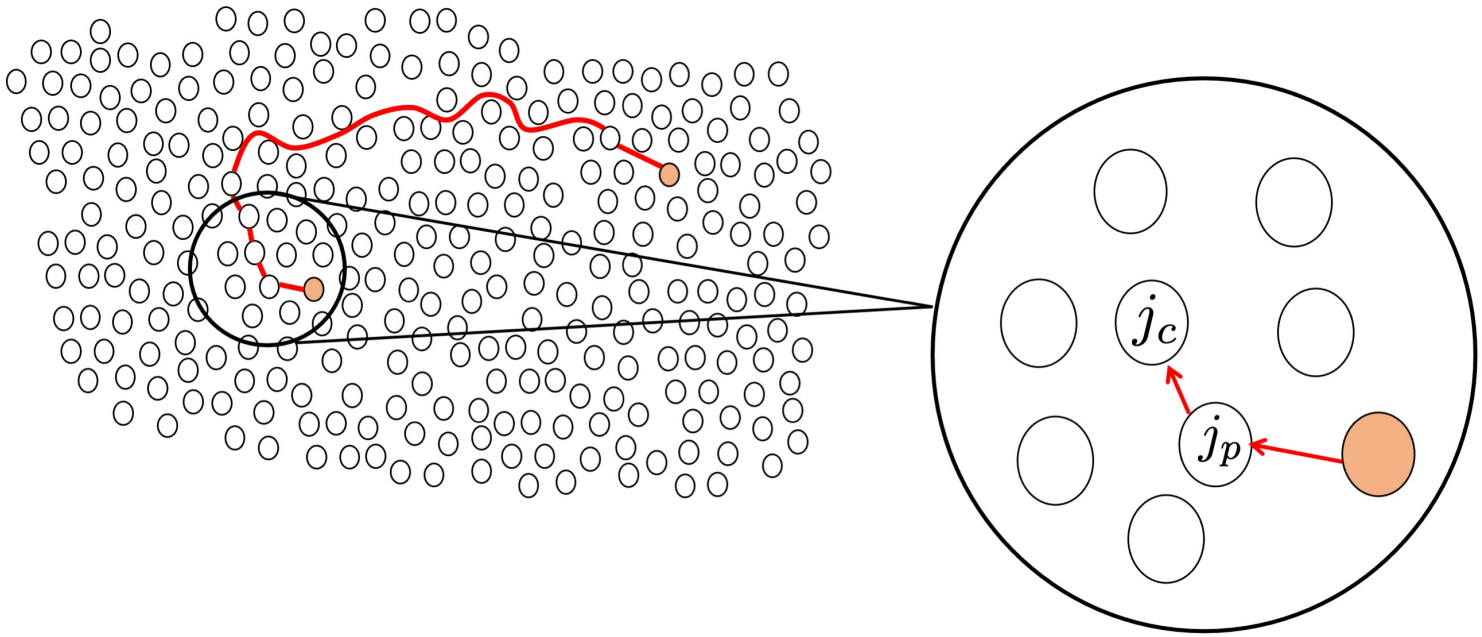
$\Phi_N(j_1, j_2) = 0 \iff j_1, j_2$ are j-invariants of N-isogenous elliptic curves.

This tells us that the roots of $\Phi_{N,p}(X, j)$ are neighbours of j in $\Gamma_1(N; p)$.

[2] Reducing coefficients mod p we can work with $\Phi_{N,p}(X, Y) \in \mathbb{F}_p[X, Y]$.

Application to Dimension One

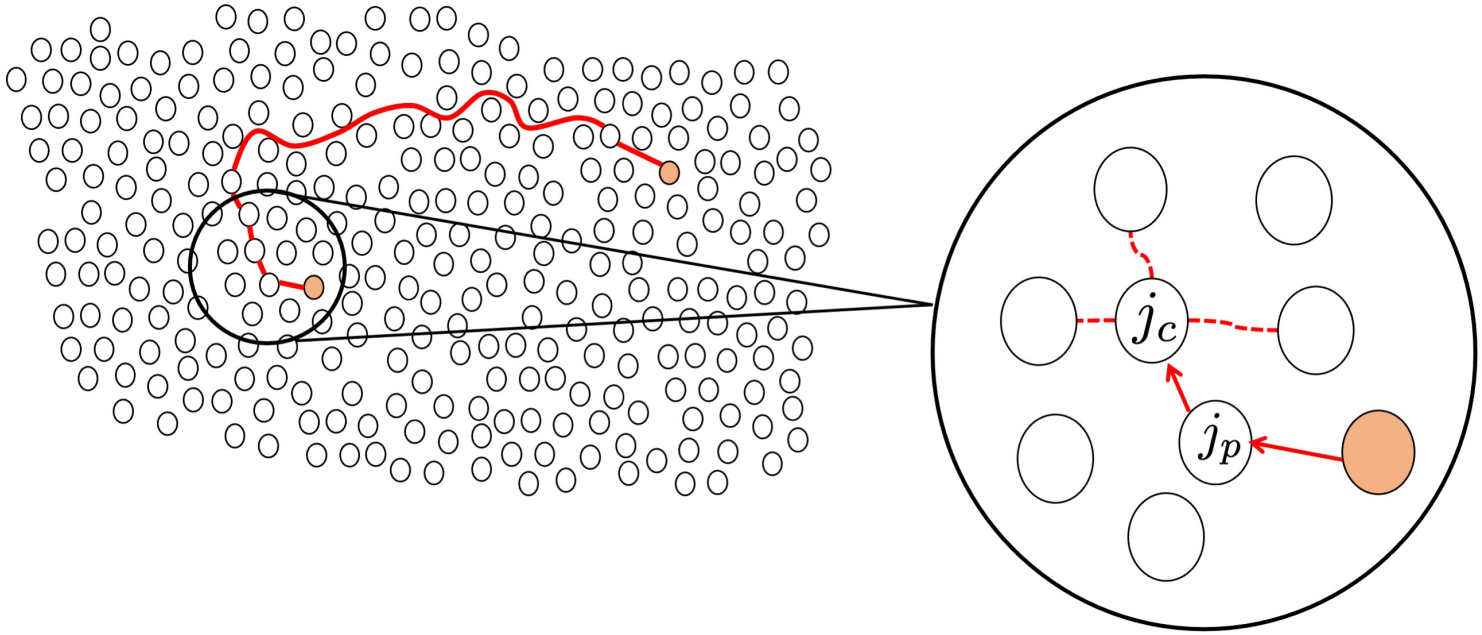
The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One: how to take steps



1. Store the current and previous j-invariants j_c and j_p.

Application to Dimension One

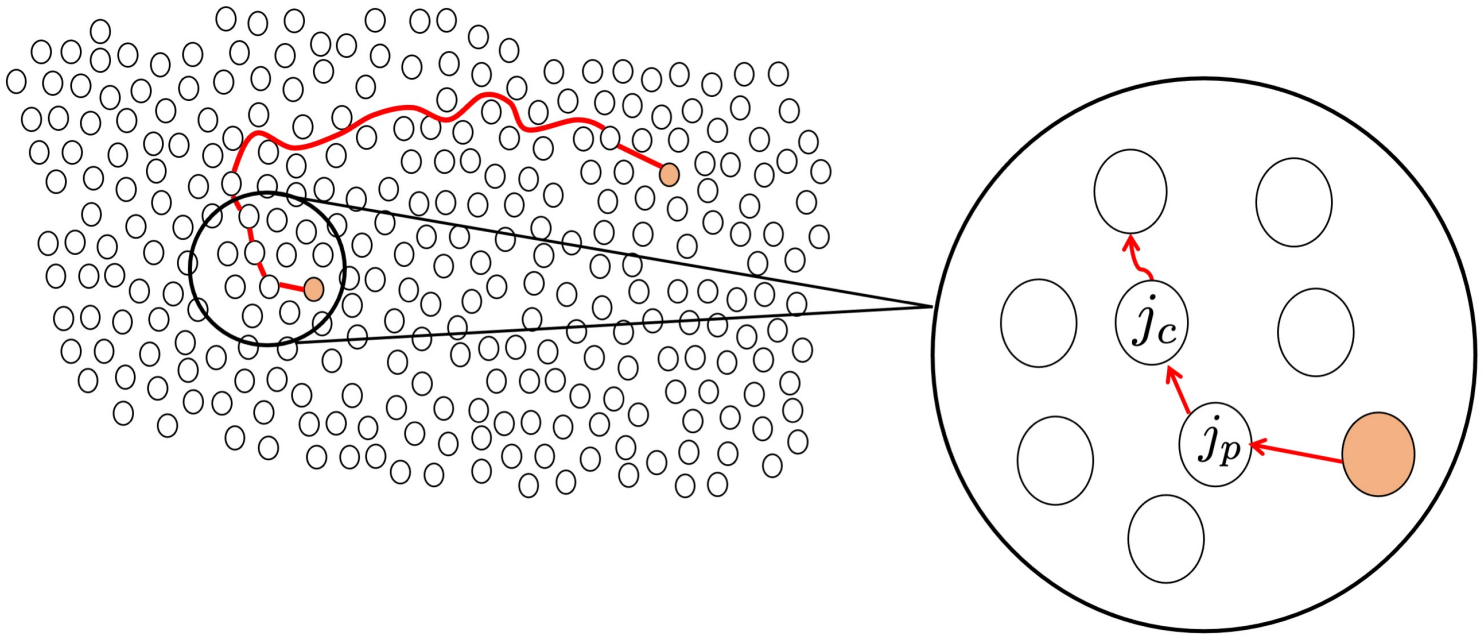
The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One: how to take steps



2. Find the $D_N - 1$ roots of $\Phi_{N,p}(X, j_c)/(X - j_p)$.

Application to Dimension One

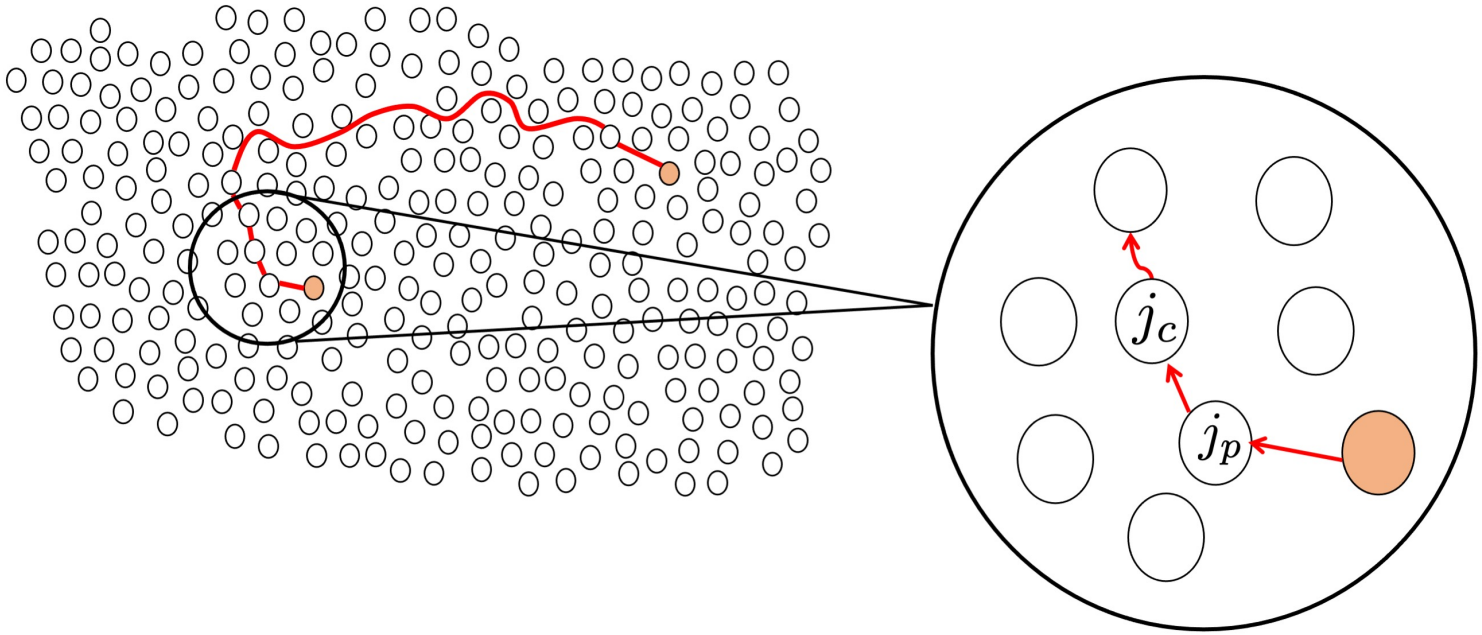
The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One: how to take steps



3. Choose one of these and walk to the corresponding node.

Application to Dimension One

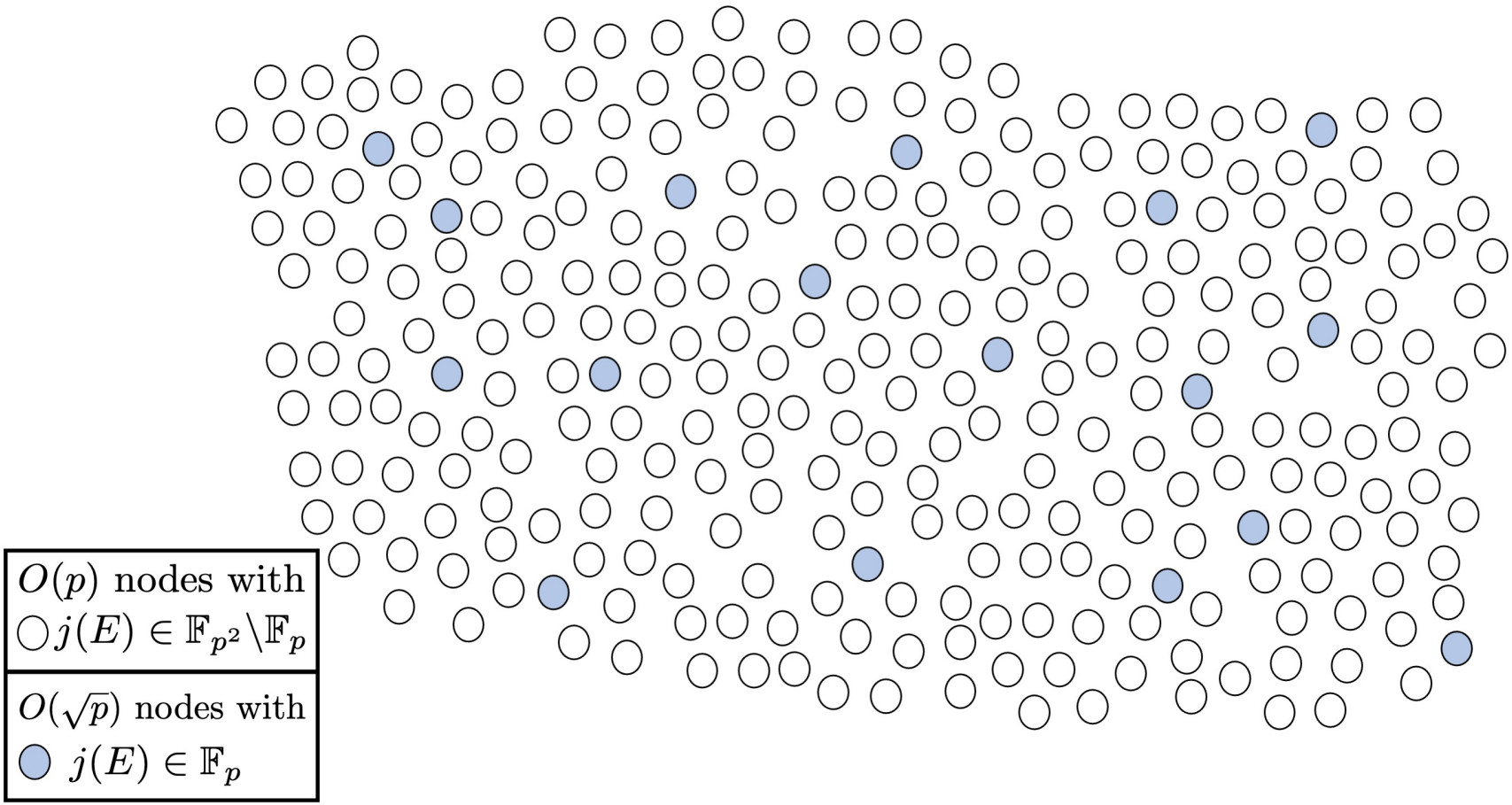
The N-Isogeny Graph $\Gamma_1(N; p)$ in Dimension One: how to take steps



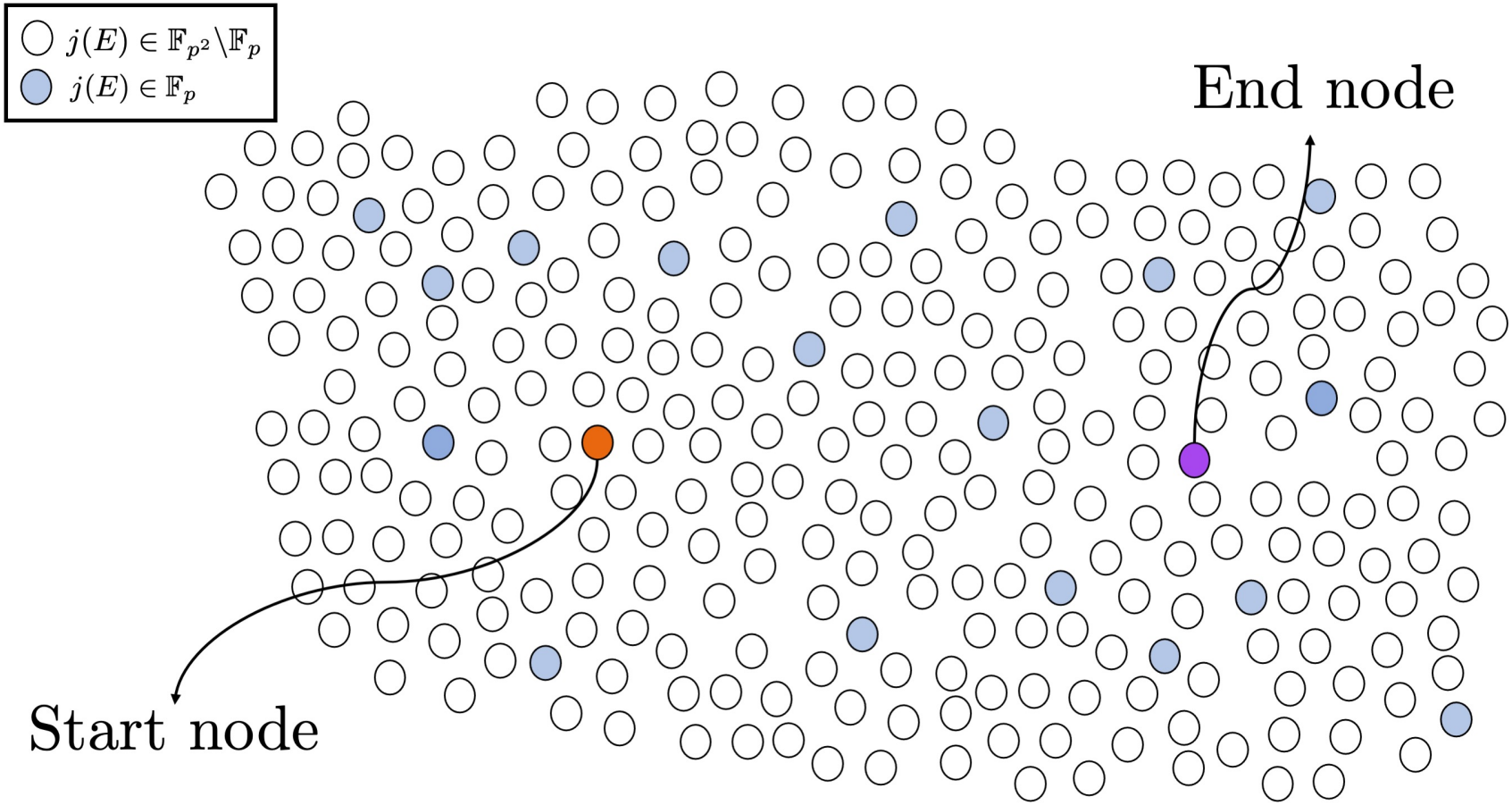
3. Choose one of these and walk to the corresponding node.

We choose to take *non-backtracking* walks in $\Gamma_1(2; p)$ as this corresponds to factoring a quadratic. So, each step reveals 2 nodes.

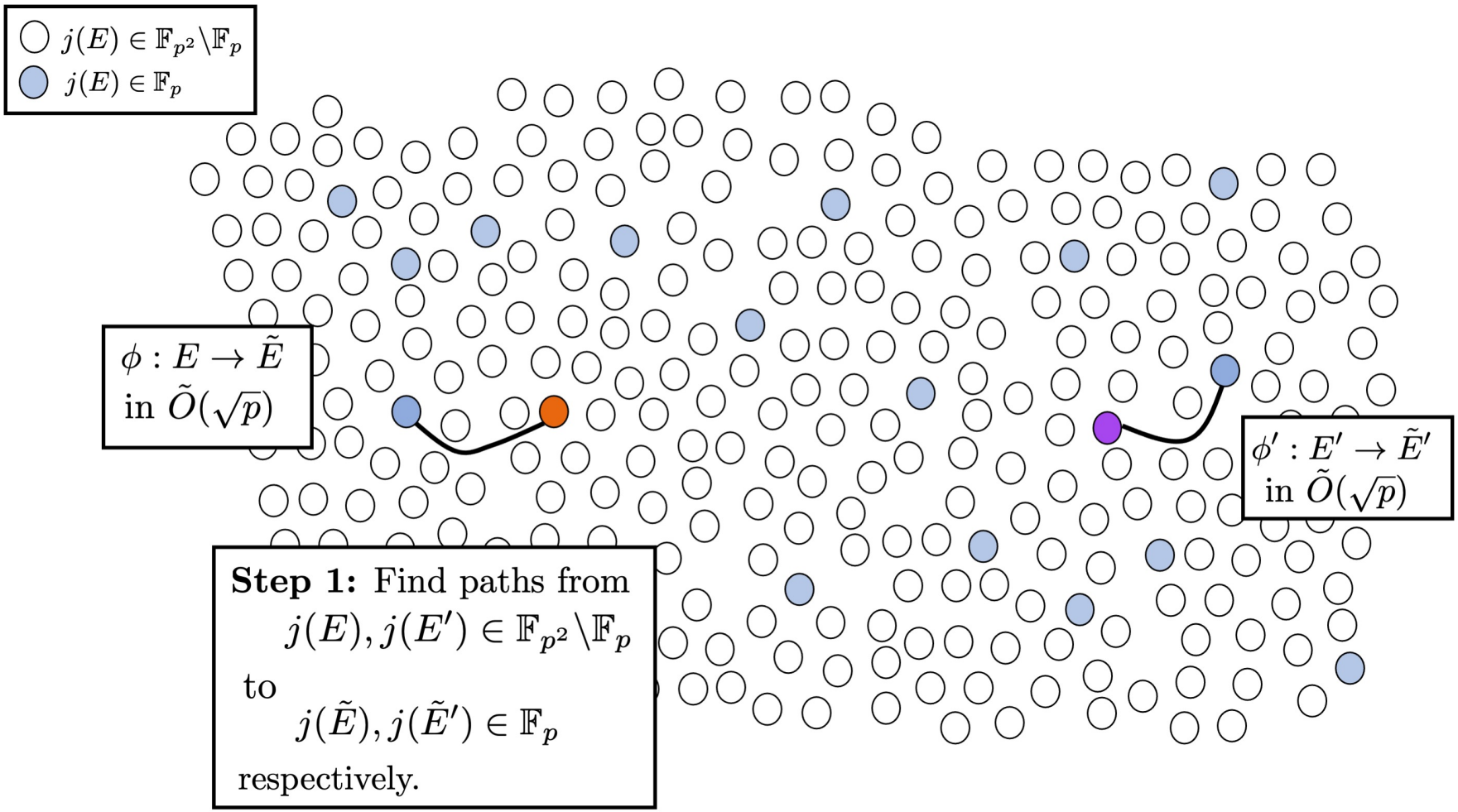
Special Subset: $\mathbb{F}_p$ nodes



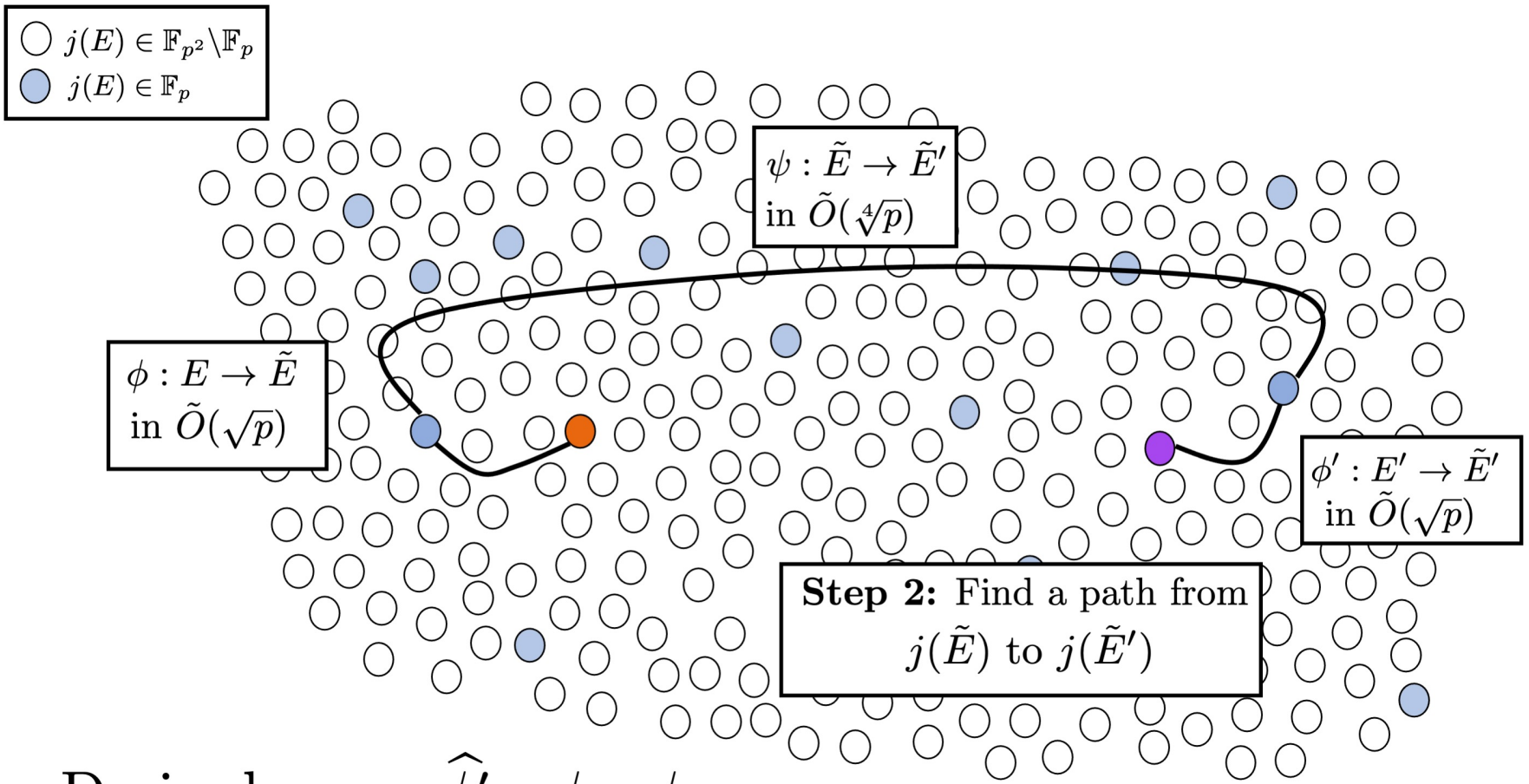
The Delfs–Galbraith Algorithm



The Delfs–Galbraith Algorithm



The Delfs–Galbraith Algorithm



Desired map: $\widehat{\phi'} \circ \psi \circ \phi$

Improved Detection of $\mathbb{F}_p$-Subgraph

For $N > 2$, we want to determine whether the current node we are on $j_c \in \mathbb{F}_{p^2}$ is N-isogenous to node in $\mathbb{F}_p$, i.e., we ask:

$$\text{Does } \Phi_{N,p}(X, j_c) \text{ have a root in } \mathbb{F}_p?$$

Improved Detection of $\mathbb{F}_p$-Subgraph

For $N > 2$, we want to determine whether the current node we are on $j_c \in \mathbb{F}_{p^2}$ is N-isogenous to node in $\mathbb{F}_p$, i.e., we ask:

$$\text{Does } \Phi_{N,p}(X, j_c) \text{ have a root in } \mathbb{F}_p?$$

Key Observation

At each step, the precise values roots do not need to be known, only whether one lies in $\mathbb{F}_p$.

Improved Detection of $\mathbb{F}_p$-Subgraph

For $N > 2$, we want to determine whether the current node we are on $j_c \in \mathbb{F}_{p^2}$ is N-isogenous to node in $\mathbb{F}_p$, i.e., we ask:

$$\text{Does } \Phi_{N,p}(X, j_c) \text{ have a root in } \mathbb{F}_p?$$

Key Observation

At each step, the precise values roots do not need to be known, only whether one lies in $\mathbb{F}_p$.

We give a fast way of detecting whether it has a root in $\mathbb{F}_p$ *without* finding roots.

Improved Detection of $\mathbb{F}_p$-Subgraph

For $N > 2$, we want to determine whether the current node we are on $j_c \in \mathbb{F}_{p^2}$ is N-isogenous to node in $\mathbb{F}_p$, i.e., we ask:

$$\text{Does } \Phi_{N,p}(X, j_c) \text{ have a root in } \mathbb{F}_p?$$

Key Observation

At each step, the precise values roots do not need to be known, only whether one lies in $\mathbb{F}_p$.

We give a fast way of detecting whether it has a root in $\mathbb{F}_p$ *without* finding roots.

(1) Evaluate $\Phi_{N,p}(X, Y)$ at $Y = j_c$.

Improved Detection of $\mathbb{F}_p$-Subgraph

For $N > 2$, we want to determine whether the current node we are on $j_c \in \mathbb{F}_{p^2}$ is N-isogenous to node in $\mathbb{F}_p$, i.e., we ask:

$$\text{Does } \Phi_{N,p}(X, j_c) \text{ have a root in } \mathbb{F}_p?$$

Key Observation

At each step, the precise values roots do not need to be known, only whether one lies in $\mathbb{F}_p$.

We give a fast way of detecting whether it has a root in $\mathbb{F}_p$ *without* finding roots.

(1) Evaluate $\Phi_{N,p}(X, Y)$ at $Y = j_c$. Let $f = \Phi_{N,p}(X, j_c)$.
(2) Compute $\gcd(f, \pi(f))$, where π is the Frobenius map.

Improved Detection of $\mathbb{F}_p$-Subgraph

For $N > 2$, we want to determine whether the current node we are on $j_c \in \mathbb{F}_{p^2}$ is N-isogenous to node in $\mathbb{F}_p$, i.e., we ask:

$$\text{Does } \Phi_{N,p}(X, j_c) \text{ have a root in } \mathbb{F}_p?$$

Key Observation

At each step, the precise values roots do not need to be known, only whether one lies in $\mathbb{F}_p$.

We give a fast way of detecting whether it has a root in $\mathbb{F}_p$ *without* finding roots.

(1) Evaluate $\Phi_{N,p}(X, Y)$ at $Y = j_c$. Let $f = \Phi_{N,p}(X, j_c)$.

(2) Compute $\gcd(f, \pi(f))$, where π is the Frobenius map.

If $\deg\big(\gcd(f, \pi(f))\big) = 1$, f has a root in $\mathbb{F}_p$.

If $\deg\big(\gcd(f, \pi(f))\big) = 0$, f does not have a root in $\mathbb{F}_p$.

Improved Detection of $\mathbb{F}_p$-Subgraph

For $N > 2$, we want to determine whether the current node we are on $j_c \in \mathbb{F}_{p^2}$ is N-isogenous to node in $\mathbb{F}_p$, i.e., we ask:

$$\text{Does } \Phi_{N,p}(X, j_c) \text{ have a root in } \mathbb{F}_p?$$

Key Observation

At each step, the precise values roots do not need to be known, only whether one lies in $\mathbb{F}_p$.

We give a fast way of detecting whether it has a root in $\mathbb{F}_p$ *without* finding roots.

(1) Evaluate $\Phi_{N,p}(X, Y)$ at $Y = j_c$. Let $f = \Phi_{N,p}(X, j_c)$.

(2) Compute $\gcd(f, \pi(f))$, where π is the Frobenius map.

If $\deg\left(\gcd(f, \pi(f))\right) = 1$, f has a root in $\mathbb{F}_p$.

If $\deg\left(\gcd(f, \pi(f))\right) = 0$, f does not have a root in $\mathbb{F}_p$.

We also show how to transform $f, \pi(f) \in \mathbb{F}_{p^2}[X]$ to give $g_1, g_2 \in \mathbb{F}_p[X]$ with the same gcd and avoid *all* costly multiplications in $\mathbb{F}_{p^2}$.

Improved Detection of $\mathbb{F}_p$-Subgraph

We give a fast way of detecting whether it has a root in $\mathbb{F}_p$ *without* finding roots.

(1) Evaluate $\Phi_{N,p}(X, Y)$ at $Y = j_c$. Let $f = \Phi_{N,p}(X, j_c)$.

(2) Compute $\gcd(f, \pi(f))$.

If $\deg\big(\gcd(f, \pi(f))\big) = 1$, f has a root in $\mathbb{F}_p$.

If $\deg\big(\gcd(f, \pi(f))\big) = 0$, f does not have a root in $\mathbb{F}_p$.

We choose the detection set $\mathcal{D} \subseteq \mathbb{Z}_{\geq 2}$ to minimise

$$\frac{\text{cost of step in graph} + \text{cost of (1) \& (2)}}{\text{nodes revealed by walk and detection}}.$$

Improved Detection of $\mathbb{F}_p$-Subgraph

We give a fast way of detecting whether it has a root in $\mathbb{F}_p$ *without* finding roots.

(1) Evaluate $\Phi_{N,p}(X, Y)$ at $Y = j_c$. Let $f = \Phi_{N,p}(X, j_c)$.

(2) Compute $\gcd(f, \pi(f))$.

If $\deg\big(\gcd(f, \pi(f))\big) = 1$, f has a root in $\mathbb{F}_p$.

If $\deg\big(\gcd(f, \pi(f))\big) = 0$, f does not have a root in $\mathbb{F}_p$.

We choose the detection set $\mathcal{D} \subseteq \mathbb{Z}_{\geq 2}$ to minimise

$$\frac{\text{cost of computing square root of } \Phi_{2,p}(X, j_c)/(X - j_p) + \text{cost of (1) \& (2)}}{2 + \text{nodes revealed by detection}}.$$

Application to Dimension Two

Based on joint with Craig Costello, Sam Frengley (eprint soon): SplitSearcher

We now consider the general isogeny problem in dimension 2, which can be viewed as finding a path between two nodes A_1, A_2 in the superspecial isogeny graph.

Application to Dimension Two

The (N, N)-Isogeny Graph $\Gamma_2(N; p)$ in Dimension Two: taking a steps

Application to Dimension Two

The (N, N)-Isogeny Graph $\Gamma_2(N; p)$ in Dimension Two: taking a steps

- To take a step from current node A_c in $\Gamma_2(N; p)$, we need to compute one of the $O(N^3)$ (N, N)-isogenies with domain A_c, say ϕ.

Application to Dimension Two

The (N, N)-Isogeny Graph $\Gamma_2(N; p)$ in Dimension Two: taking a steps

- To take a step from current node A_c in $\Gamma_2(N; p)$, we need to compute one of the $O(N^3)$ (N, N)-isogenies with domain A_c, say ϕ.
- We choose $N = 2$ to exploit fast Richelot isogenies

Application to Dimension Two

The (N, N)-Isogeny Graph $\Gamma_2(N; p)$ in Dimension Two: taking a steps

- To take a step from current node A_c in $\Gamma_2(N; p)$, we need to compute one of the $O(N^3)$ (N, N)-isogenies with domain A_c, say ϕ.
- We choose $N = 2$ to exploit fast Richelot isogenies *and* expansion properties of random walks are most well understood in $\Gamma_2(2; p)$.

Application to Dimension Two

The (N, N)-Isogeny Graph $\Gamma_2(N; p)$ in Dimension Two: taking a steps

- To take a step from current node A_c in $\Gamma_2(N; p)$, we need to compute one of the $O(N^3)$ (N, N)-isogenies with domain A_c, say ϕ.
- We choose $N = 2$ to exploit fast Richelot isogenies *and* expansion properties of random walks are most well understood in $\Gamma_2(2; p)$.
- For each node A_c, there are 15 outgoing $(2, 2)$-isogenies but we restrict to only 8 *good extensions*.

Application to Dimension Two

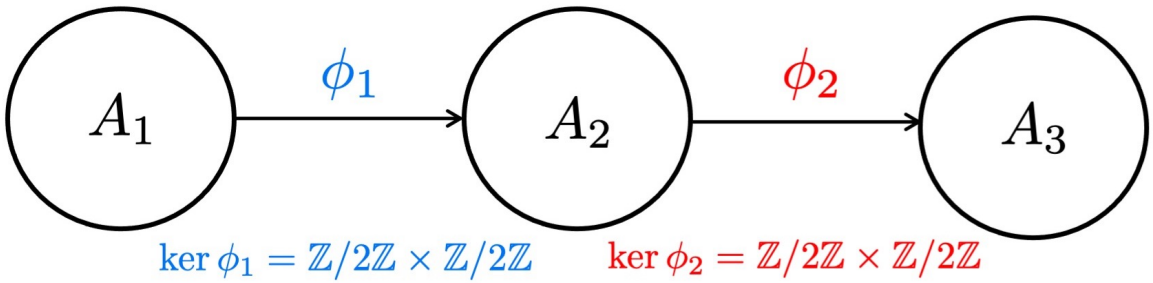
The (N, N)-Isogeny Graph $\Gamma_2(N; p)$ in Dimension Two: taking a steps

- To take a step from current node A_c in $\Gamma_2(N; p)$, we need to compute one of the $O(N^3)$ (N, N)-isogenies with domain A_c, say ϕ.
- We choose $N = 2$ to exploit fast Richelot isogenies *and* expansion properties of random walks are most well understood in $\Gamma_2(2; p)$.
- For each node A_c, there are 15 outgoing $(2, 2)$-isogenies but we restrict to only 8 *good extensions*. These ensure that there are no trivial cycles.

Application to Dimension Two

The (N, N)-Isogeny Graph $\Gamma_2(N; p)$ in Dimension Two: taking a steps

- To take a step from current node A_c in $\Gamma_2(N; p)$, we need to compute one of the $O(N^3)$ (N, N)-isogenies with domain A_c, say ϕ.
- We choose $N = 2$ to exploit fast Richelot isogenies *and* expansion properties of random walks are most well understood in $\Gamma_2(2; p)$.
- For each node A_c, there are 15 outgoing $(2, 2)$-isogenies but we restrict to only 8 *good extensions*. These ensure that there are no trivial cycles.



$$\ker \hat{\phi}_1 \cap \ker \phi_2 = \begin{cases} \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} & \phi_2 \text{ is dual to } \phi_1 \\ \mathbb{Z}/2\mathbb{Z} & \phi_2 \text{ is a bad extension of } \phi_1 \\ \emptyset & \phi_2 \text{ is a good extension of } \phi_1 \end{cases}$$

Special Subset: Product nodes

Every p.p. abelian surface is isomorphic to either the Jacobian of a curve of genus 2, or to a product of two elliptic curves.

Special Subset: Product nodes

Every p.p. abelian surface is isomorphic to either the Jacobian of a curve of genus 2, or to a product of two elliptic curves.
So, the vertex set $\mathcal{S}(p)$ of $\Gamma_2(N; p)$ is equal to the disjoint union of:

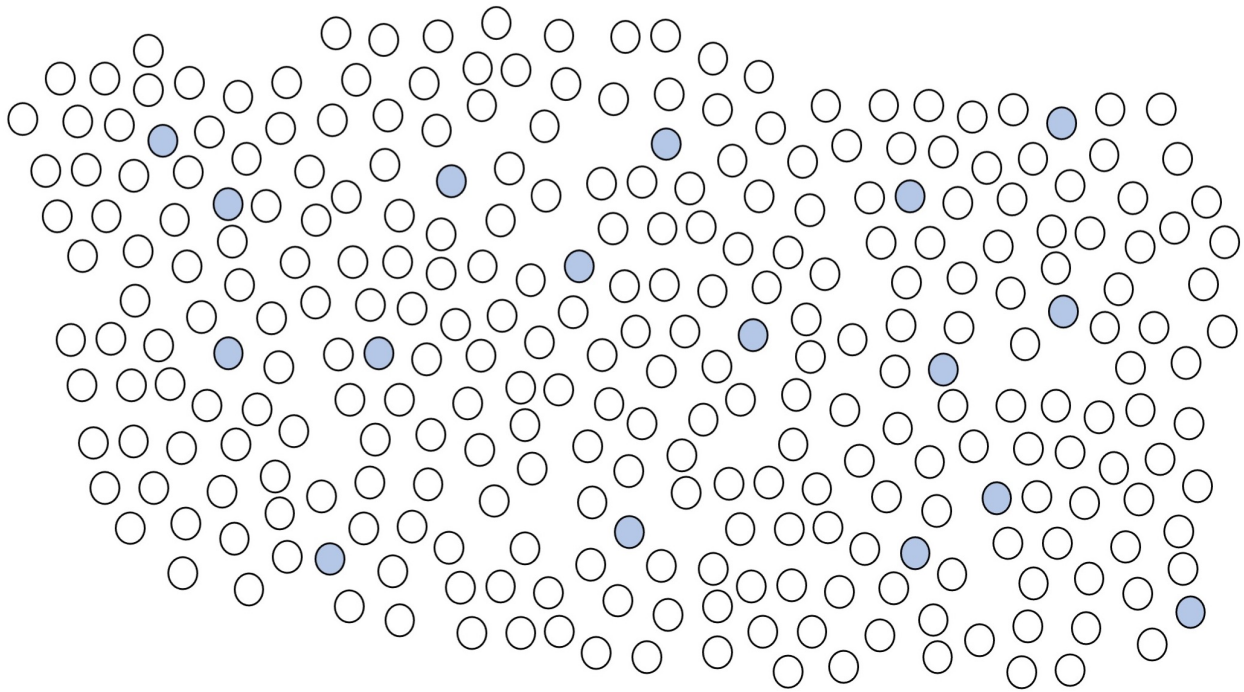
$$\mathcal{J}(p) := \{[A] \in \mathcal{S}(p) \ : \ A \cong \mathrm{Jac}(C)\} \text{ and}$$
$$\mathcal{E}(p) := \{[A] \in \mathcal{S}(p) \ : \ A \cong E \times E' \text{ with } E,\ E' \text{ supersingular ECs}\}.$$

Special Subset: Product nodes

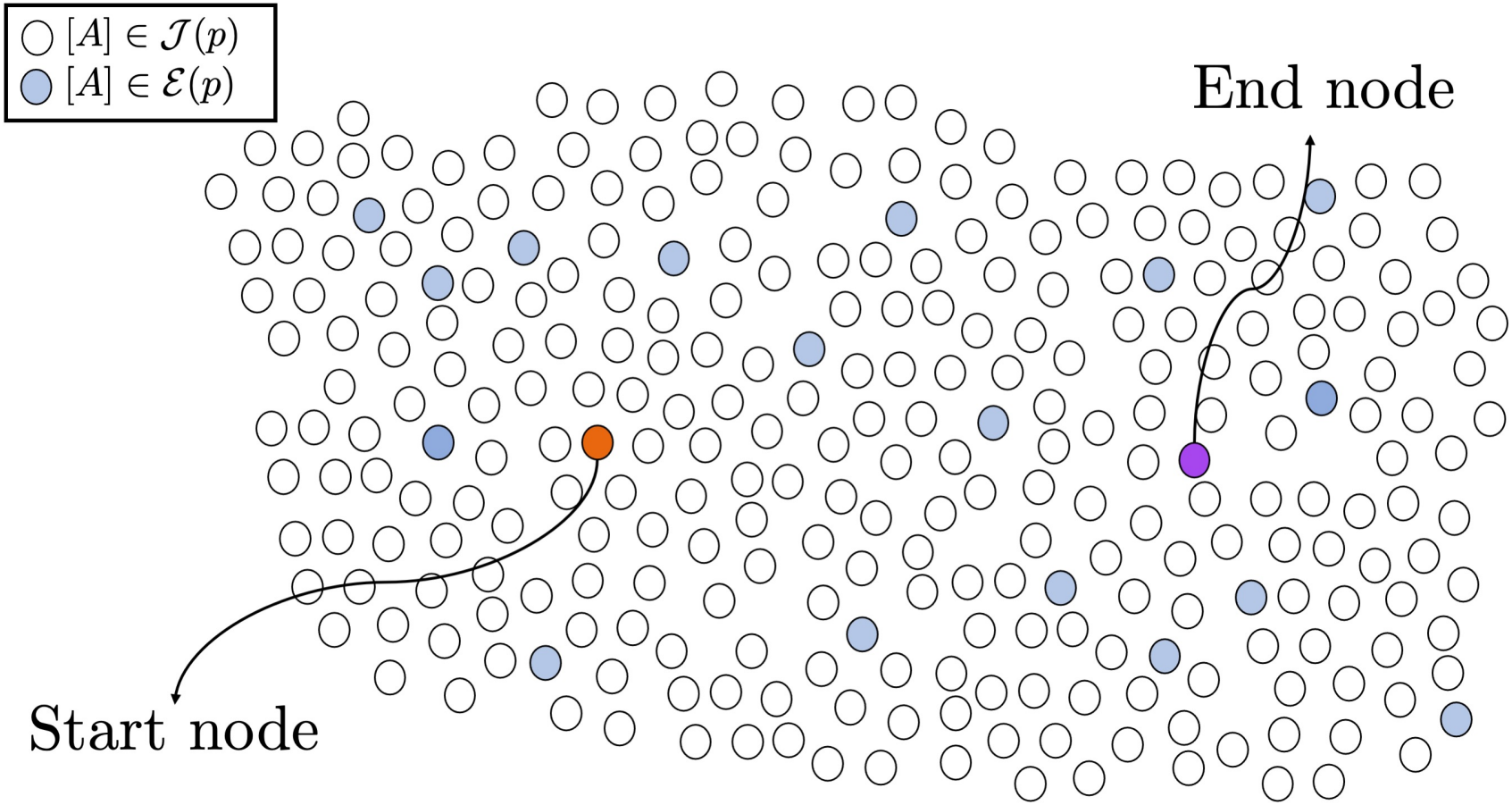
Every p.p. abelian surface is isomorphic to either the Jacobian of a curve of genus 2, or to a product of two elliptic curves.
So, the vertex set $\mathcal{S}(p)$ of $\Gamma_2(N; p)$ is equal to the disjoint union of:

$$\mathcal{J}(p) := \{[A] \in \mathcal{S}(p) \ : \ A \cong \mathrm{Jac}(C)\} \text{ and}$$
$$\mathcal{E}(p) := \{[A] \in \mathcal{S}(p) \ : \ A \cong E \times E' \text{ with } E,\ E' \text{ supersingular ECs}\}.$$

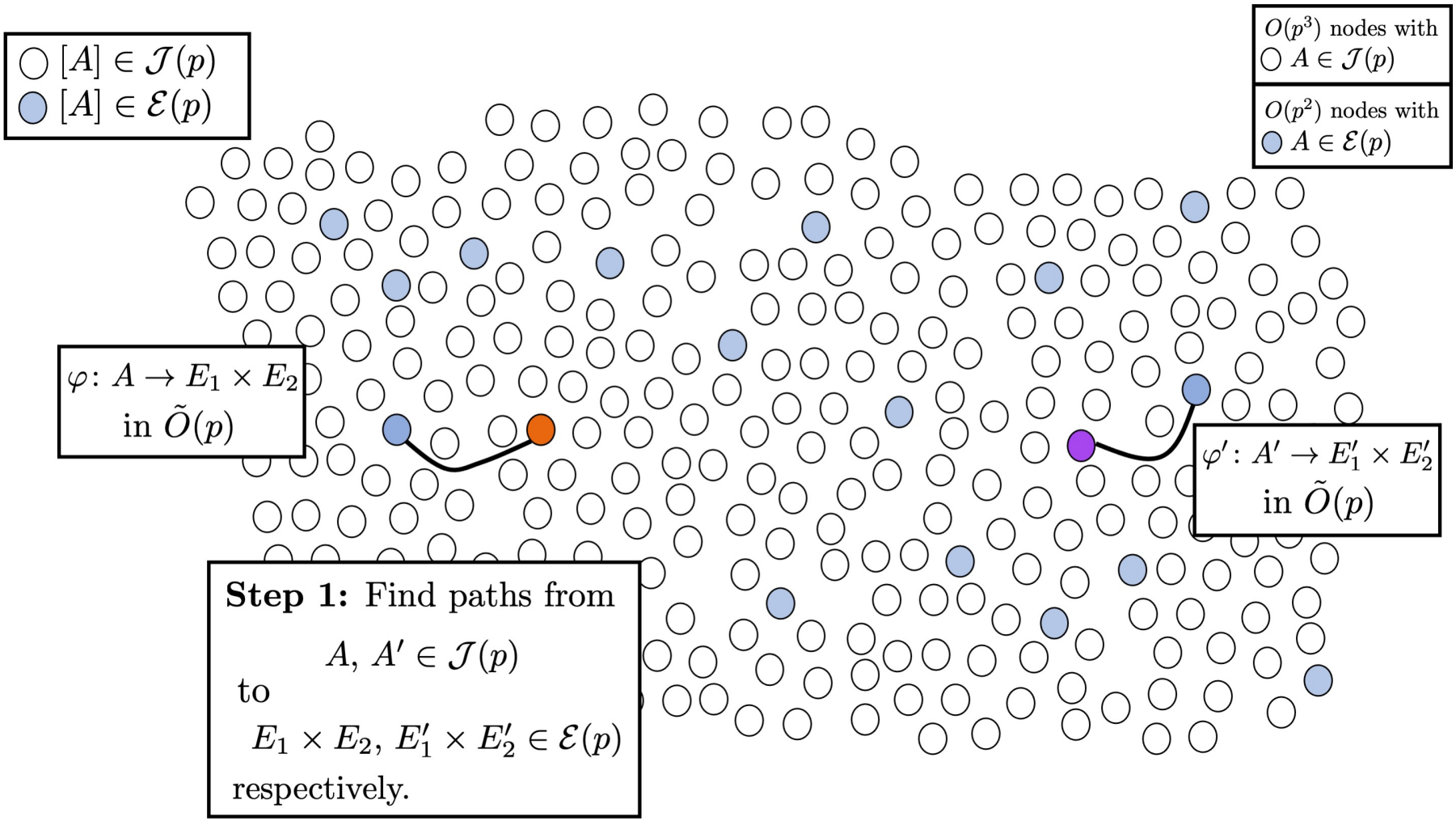
$O(p^3)$ nodes with ○ $A \in \mathcal{J}(p)$
$O(p^2)$ nodes with ● $A \in \mathcal{E}(p)$



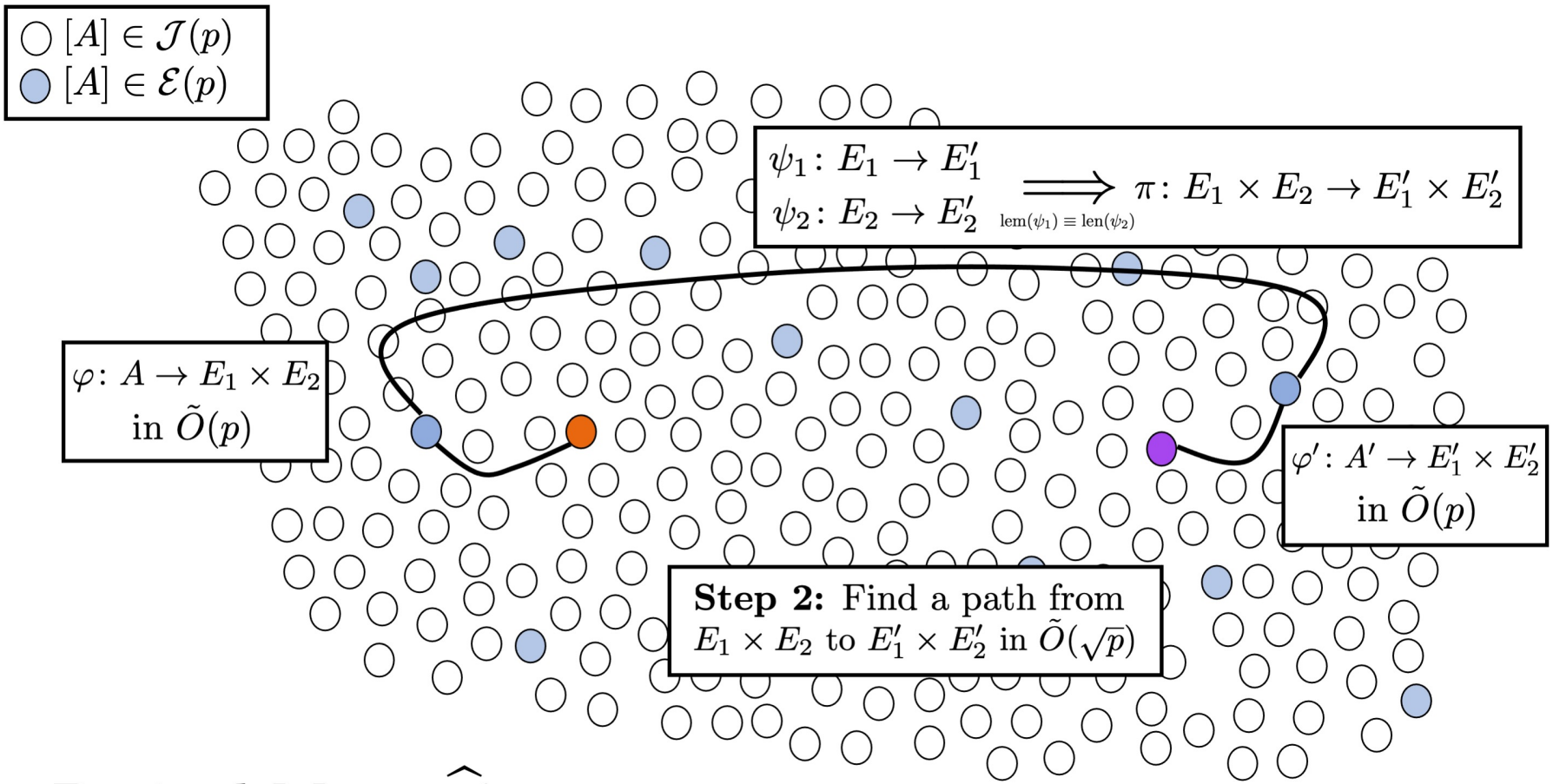
The Costello–Smith Algorithm



The Costello–Smith Algorithm



The Costello–Smith Algorithm



Desired Map: $\widehat{\varphi'} \circ \pi \circ \varphi$

Improved Detection of Product Subgraph

The bottleneck of the attack is the first step: walking in $\Gamma_2(2; p)$ until finding $A \in \mathcal{J}(p)$ which is $(2, 2)$-*split*.

Improved Detection of Product Subgraph

The bottleneck of the attack is the first step: walking in $\Gamma_2(2; p)$ until finding $A \in \mathcal{J}(p)$ which is $(2, 2)$-*split*.

Definition

We say the Jacobian $\mathrm{Jac}(C)$ of a genus 2 curve C is (N, N)-*split* if there exists an (N, N)-isogeny[a] $\mathrm{Jac}(C) \to E \times E'$, where E, E' are elliptic curves.

[a] Separable, polarised, optimal

Improved Detection of Product Subgraph

The bottleneck of the attack is the first step: walking in $\Gamma_2(2; p)$ until finding $A \in \mathcal{J}(p)$ which is $(2, 2)$-*split*.

Definition

We say the Jacobian $\mathrm{Jac}(C)$ of a genus 2 curve C is (N, N)-*split* if there exists an (N, N)-isogeny[a] $\mathrm{Jac}(C) \to E \times E'$, where E, E' are elliptic curves.

[a] Separable, polarised, optimal

First step in more detail:

1. We start on a node $A_0 \in \mathcal{J}(p)$.

Improved Detection of Product Subgraph

The bottleneck of the attack is the first step: walking in $\Gamma_2(2; p)$ until finding $A \in \mathcal{J}(p)$ which is $(2, 2)$-*split*.

Definition

We say the Jacobian $\mathrm{Jac}(C)$ of a genus 2 curve C is (N, N)-*split* if there exists an (N, N)-isogeny[a] $\mathrm{Jac}(C) \to E \times E'$, where E, E' are elliptic curves.

[a] Separable, polarised, optimal

First step in more detail:

1. We start on a node $A_0 \in \mathcal{J}(p)$.
2. Take a step in $\Gamma_2(2; p)$ via a Richelot isogeny $\phi_1 : A_0 \to A_1$.

Improved Detection of Product Subgraph

The bottleneck of the attack is the first step: walking in $\Gamma_2(2; p)$ until finding $A \in \mathcal{J}(p)$ which is $(2, 2)$-*split*.

Definition

We say the Jacobian $\mathrm{Jac}(C)$ of a genus 2 curve C is (N, N)-*split* if there exists an (N, N)-isogeny[a] $\mathrm{Jac}(C) \to E \times E'$, where E, E' are elliptic curves.

[a] Separable, polarised, optimal

First step in more detail:

1. We start on a node $A_0 \in \mathcal{J}(p)$.
2. Take a step in $\Gamma_2(2; p)$ via a Richelot isogeny $\phi_1 : A_0 \to A_1$.
3. From the Richelot isogeny formulae, we can determine whether $A_1 \in \mathcal{E}(p)$. If not, take another step $\phi_2 : A_1 \to A_2$.

Improved Detection of Product Subgraph

The bottleneck of the attack is the first step: walking in $\Gamma_2(2; p)$ until finding $A \in \mathcal{J}(p)$ which is $(2, 2)$-*split*.

Definition

We say the Jacobian $\mathrm{Jac}(C)$ of a genus 2 curve C is (N, N)-*split* if there exists an (N, N)-isogeny[a] $\mathrm{Jac}(C) \to E \times E'$, where E, E' are elliptic curves.

[a] Separable, polarised, optimal

First step in more detail:

1. We start on a node $A_0 \in \mathcal{J}(p)$.
2. Take a step in $\Gamma_2(2; p)$ via a Richelot isogeny $\phi_1 : A_0 \to A_1$.
3. From the Richelot isogeny formulae, we can determine whether $A_1 \in \mathcal{E}(p)$. If not, take another step $\phi_2 : A_1 \to A_2$.
4. Repeat previous step until finding $A_i \in \mathcal{E}(p)$.

Improved Detection of Product Subgraph

The bottleneck of the attack is the first step: walking in $\Gamma_2(2; p)$ until finding $A \in \mathcal{J}(p)$ which is $(2, 2)$-*split*.

Definition

We say the Jacobian $\mathrm{Jac}(C)$ of a genus 2 curve C is (N, N)-*split* if there exists an (N, N)-isogeny[a] $\mathrm{Jac}(C) \to E \times E'$, where E, E' are elliptic curves.

[a] Separable, polarised, optimal

First step in more detail:

1. We start on a node $A_0 \in \mathcal{J}(p)$.
2. Take a step in $\Gamma_2(2; p)$ via a Richelot isogeny $\phi_1 : A_0 \to A_1$.
3. From the Richelot isogeny formulae, we can determine whether $A_1 \in \mathcal{E}(p)$. If not, take another step $\phi_2 : A_1 \to A_2$.
4. Repeat previous step until finding $A_i \in \mathcal{E}(p)$.

Question: Taking steps in $\Gamma(2; p)$, can we detect whether the current node A_c is in (N, N)-split for $N > 2$?

Improved Detection of Product Subgraph

The bottleneck of the attack is the first step: walking in $\Gamma_2(2; p)$ until finding $A \in \mathcal{J}(p)$ which is $(2, 2)$-*split*.

Definition

We say the Jacobian $\mathrm{Jac}(C)$ of a genus 2 curve C is (N, N)-*split* if there exists an (N, N)-isogeny[a] $\mathrm{Jac}(C) \to E \times E'$, where E, E' are elliptic curves.

[a] Separable, polarised, optimal

First step in more detail:

1. We start on a node $A_0 \in \mathcal{J}(p)$.
2. Take a step in $\Gamma_2(2; p)$ via a Richelot isogeny $\phi_1 : A_0 \to A_1$.
3. From the Richelot isogeny formulae, we can determine whether $A_1 \in \mathcal{E}(p)$. If not, take another step $\phi_2 : A_1 \to A_2$.
4. Repeat previous step until finding $A_i \in \mathcal{E}(p)$.

Question: Taking steps in $\Gamma(2; p)$, can we detect whether the current node A_c is in (N, N)-split for $N > 2$?

Improved Detection of Product Subgraph

A first try would be to compute all (N, N)-isogenies from A_c, but this is not efficient.

Improved Detection of Product Subgraph

A first try would be to compute all (N, N)-isogenies from A_c, but this is not efficient. How can we make the detection efficient?

Improved Detection of Product Subgraph

A first try would be to compute all (N, N)-isogenies from A_c, but this is not efficient. How can we make the detection efficient?

Let $A_c = \mathrm{Jac}(C) \in \mathcal{J}(p)$ and $I_2(C), I_4(C), I_6(C), I_{10}(C)$ the Igusa–Clebsch invariants of C.

Improved Detection of Product Subgraph

A first try would be to compute all (N, N)-isogenies from A_c, but this is not efficient. How can we make the detection efficient?

Let $A_c = \mathrm{Jac}(C) \in \mathcal{J}(p)$ and $I_2(C), I_4(C), I_6(C), I_{10}(C)$ the Igusa–Clebsch invariants of C.

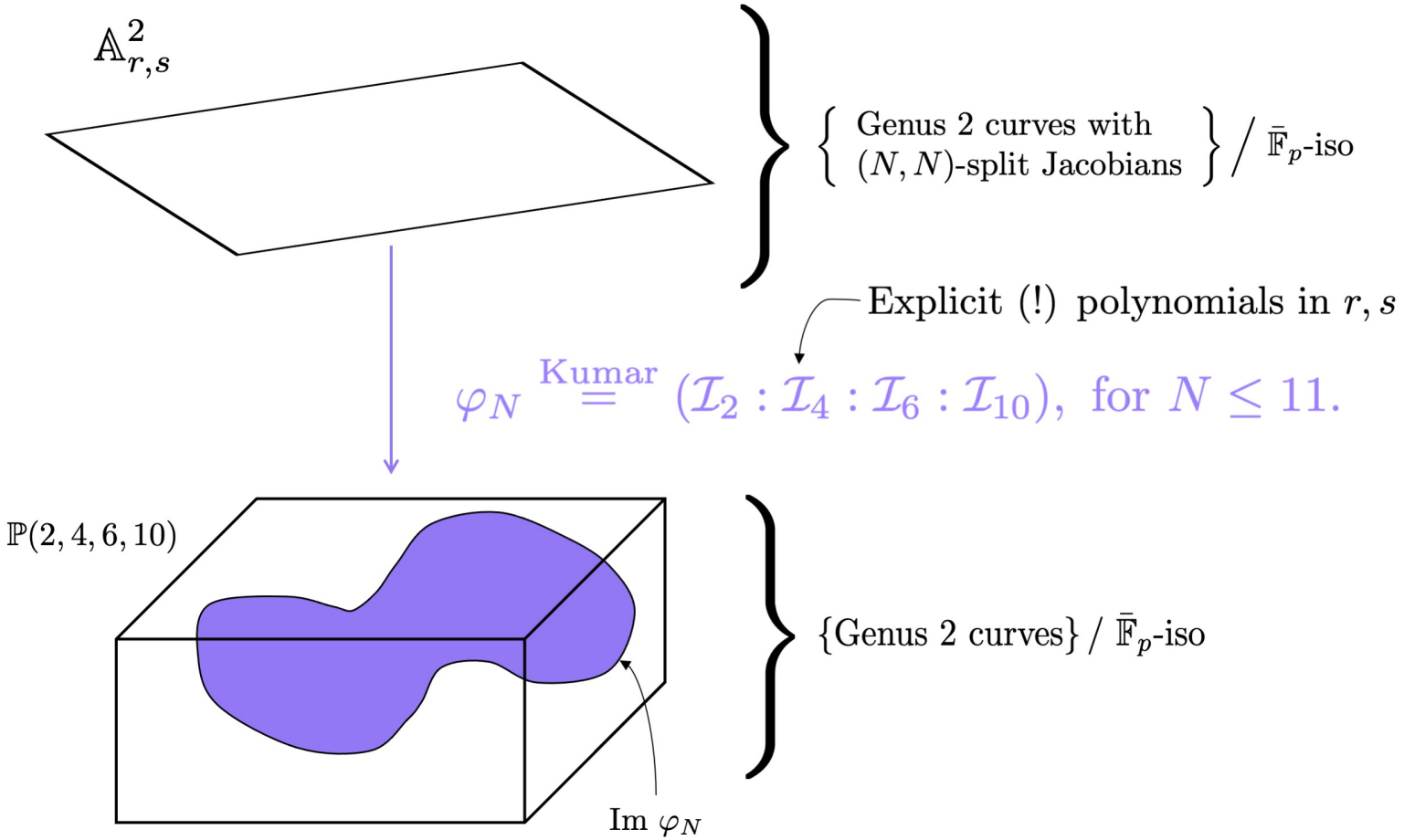
Kumar [Kum15] gives us the map

$$\varphi_N = \Big(\mathcal{I}_2(r,s) : \mathcal{I}_4(r,s) : \mathcal{I}_6(r,s) : \mathcal{I}_{10}(r,s)\Big),$$

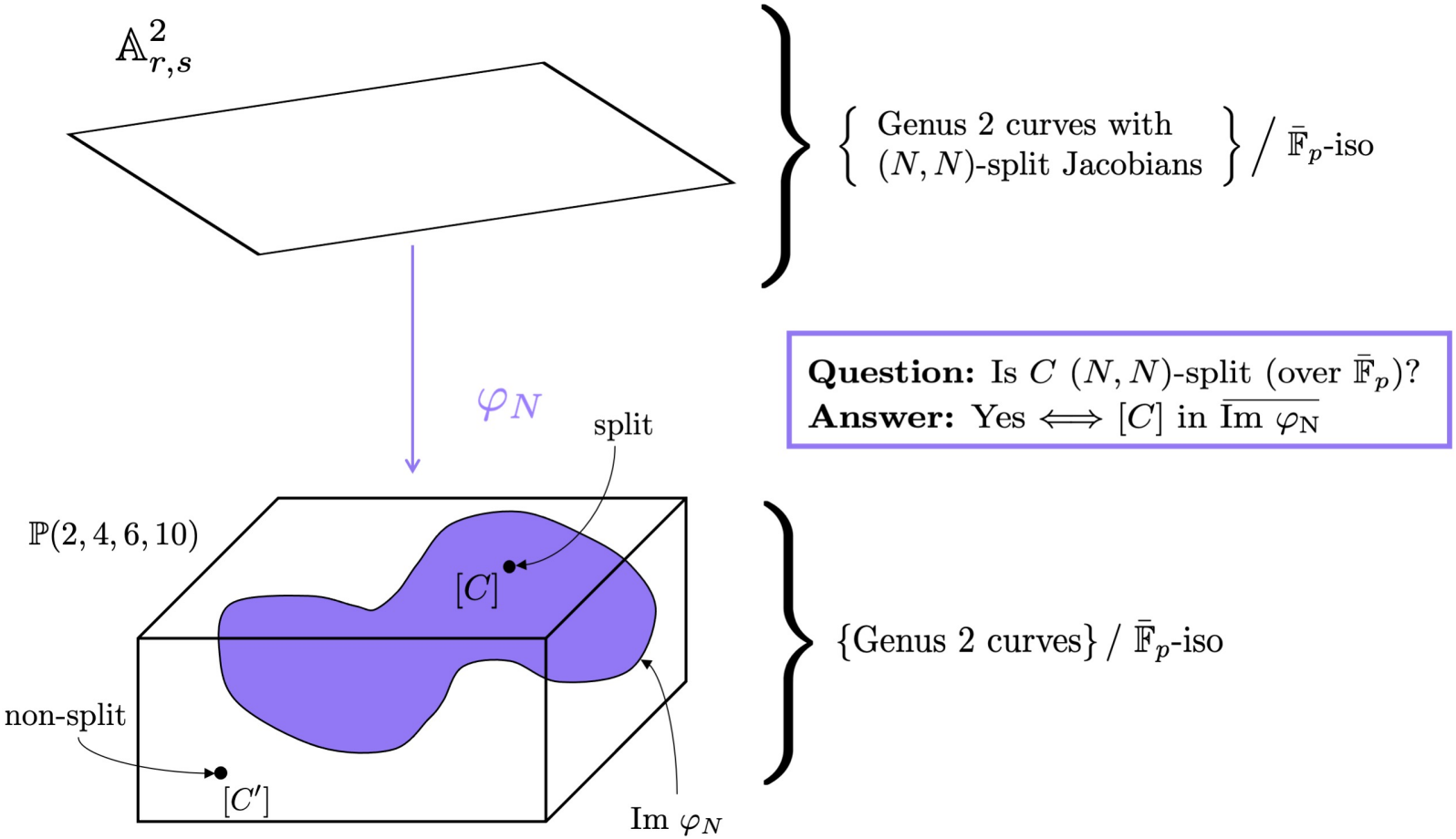
such that

$$\mathrm{Jac}(C) \text{ is } (N,N)\text{-split (over } \bar{\mathbb{F}}_p) \iff [C] \text{ in } \overline{\mathrm{Im}\ \varphi_N}$$

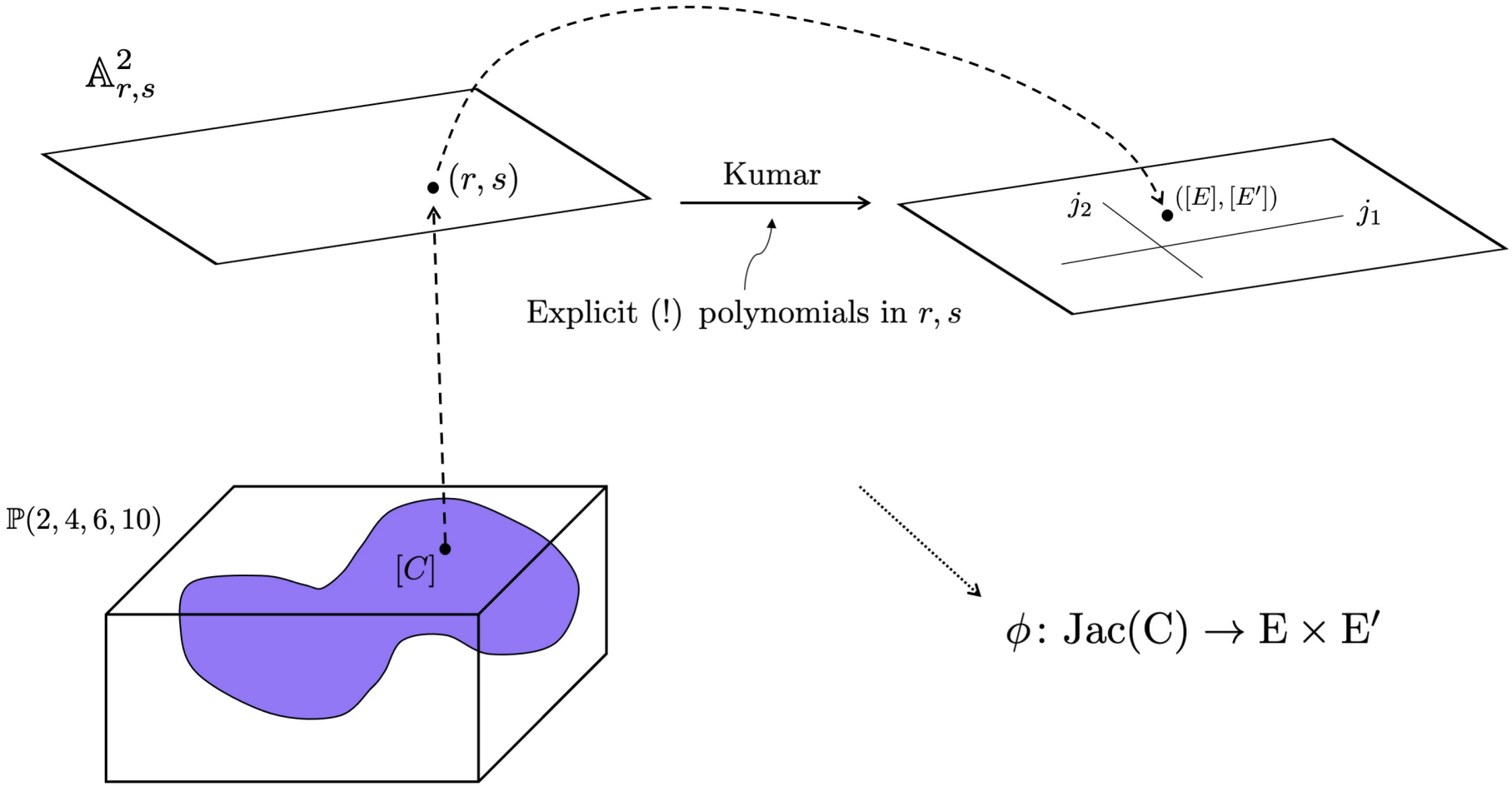
Improved Detection of Product Subgraph



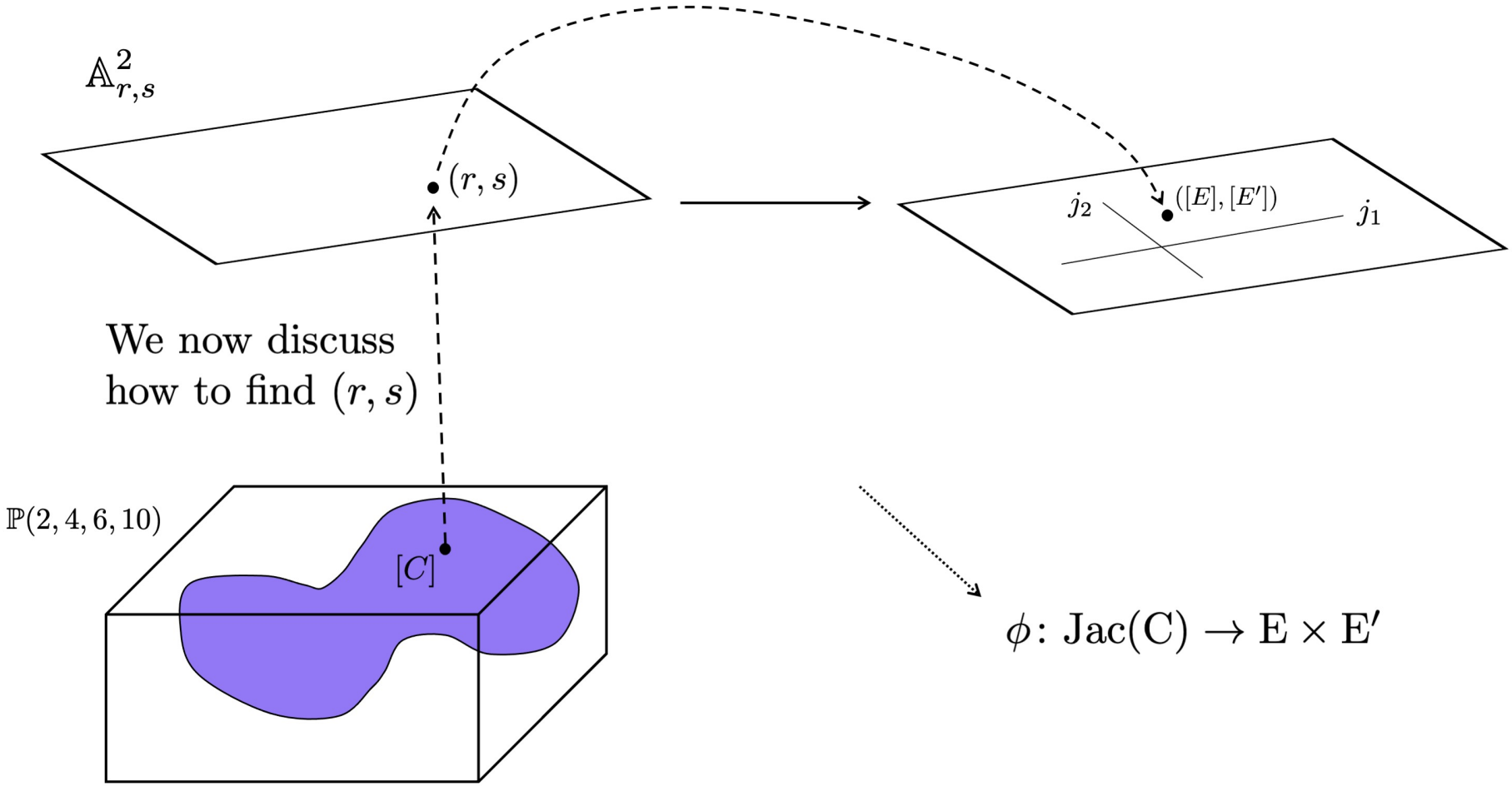
Improved Detection of Product Subgraph



Improved Detection of Product Subgraph



Improved Detection of Product Subgraph



Improved Detection of Product Subgraph

Determining if $A_c = \mathrm{Jac}(C)$ is (N, N)-isogenous to a node in $\mathcal{E}(p)$ reduces to knowing whether C lies in the image of φ_N.

Improved Detection of Product Subgraph

Determining if $A_c = \mathrm{Jac}(C)$ is (N, N)-isogenous to a node in $\mathcal{E}(p)$ reduces to knowing whether C lies in the image of φ_N. To remove projective issues, we normalise invariants as:

$$\alpha_1(C) = \frac{I_4(C)}{I_2(C)^2}, \quad \alpha_2(C) = \frac{I_2(C)I_4(C)}{I_6(C)}, \quad \alpha_3(C) = \frac{I_4(C)I_6(C)}{I_{10}(C)}$$

Improved Detection of Product Subgraph

Determining if $A_c = \mathrm{Jac}(C)$ is (N, N)-isogenous to a node in $\mathcal{E}(p)$ reduces to knowing whether C lies in the image of φ_N. To remove projective issues, we normalise invariants as:

$$\alpha_1(C) = \frac{I_4(C)}{I_2(C)^2}, \quad \alpha_2(C) = \frac{I_2(C)I_4(C)}{I_6(C)}, \quad \alpha_3(C) = \frac{I_4(C)I_6(C)}{I_{10}(C)}$$

Using the same normalisation for coordinates of Kumar's map $\mathcal{I}_k(r, s)$, we get $i_1(r, s)$, $i_2(r, s)$ and $i_3(r, s)$.

Improved Detection of Product Subgraph

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then $\mathrm{Jac}(C)$ is (N,N)-split.

Improved Detection of Product Subgraph

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then $\mathrm{Jac}(C)$ is (N,N)-split.

We determine if $\exists r_0, s_0$ by:

Improved Detection of Product Subgraph

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then $\mathrm{Jac}(C)$ is (N,N)-split.

We determine if $\exists r_0, s_0$ by:

(1) Computing resultants of (the numerators of) $f_1(r,s)$, $f_2(r,s)$ and $f_2(r,s)$, $f_3(r,s)$ (with respect to r) to get $\mathrm{res}_1(s), \mathrm{res}_2(s)$.

Improved Detection of Product Subgraph

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then $\mathrm{Jac}(C)$ is (N,N)-split.

We determine if $\exists r_0, s_0$ by:

(1) Computing resultants of (the numerators of) $f_1(r,s)$, $f_2(r,s)$ and $f_2(r,s)$, $f_3(r,s)$ (with respect to r) to get $\mathrm{res}_1(s), \mathrm{res}_2(s)$.

(2) Compute $\gcd(\mathrm{res}_1(s), \mathrm{res}_2(s))$. If degree is 0, then Jac(C) is not (N,N)-split.

Improved Detection of Product Subgraph

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then $\mathrm{Jac}(C)$ is (N,N)-split.

We determine if $\exists r_0, s_0$ by:

(1) Computing resultants of (the numerators of) $f_1(r,s)$, $f_2(r,s)$ and $f_2(r,s)$, $f_3(r,s)$ (with respect to r) to get $\mathrm{res}_1(s), \mathrm{res}_2(s)$.

(2) Compute $\gcd(\mathrm{res}_1(s), \mathrm{res}_2(s))$. If degree is 0, then Jac(C) is not (N,N)-split. Otherwise, Jac(C) is (N,N)-split and s_0 is a root of the GCD. Then solve for r_0.

Improved Detection of Product Subgraph

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then $\mathrm{Jac}(C)$ is (N,N)-split.

We determine if $\exists r_0, s_0$ by:

(1) Computing resultants of (the numerators of) $f_1(r,s)$, $f_2(r,s)$ and $f_2(r,s)$, $f_3(r,s)$ (with respect to r) to get $\mathrm{res}_1(s), \mathrm{res}_2(s)$.

(2) Compute $\gcd(\mathrm{res}_1(s), \mathrm{res}_2(s))$. If degree is 0, then Jac(C) is not (N,N)-split. Otherwise, Jac(C) is (N,N)-split and s_0 is a root of the GCD. Then solve for r_0.

In fact, we obtain a more efficient method by precomputing the resultants generically.

Improved Detection of Product Subgraph

We determine if $\exists r_0, s_0$ by:

(1) Computing resultants of (the numerators of) $f_1(r,s)$, $f_2(r,s)$ and $f_2(r,s)$, $f_3(r,s)$ (with respect to r) to get $\mathrm{res}_1(s), \mathrm{res}_2(s)$.

(2) Compute $\gcd(\mathrm{res}_1(s), \mathrm{res}_2(s))$. If degree is 0, then Jac(C) is not (N,N)-split. Otherwise, Jac(C) is (N,N)-split and s_0 is a root of the GCD. Then solve for r_0.

In fact, we obtain a more efficient method by precomputing the resultants generically.

We choose the detection set $\mathcal{D} \subseteq \{2, 3, \ldots, 11\}$ to minimise

$$\frac{\text{cost of step in graph} + \text{cost of (1) \& (2)}}{\text{nodes revealed by walk and detection}}.$$

Improved Detection of Product Subgraph

We determine if $\exists r_0, s_0$ by:

(1) Computing resultants of (the numerators of) $f_1(r,s)$, $f_2(r,s)$ and $f_2(r,s)$, $f_3(r,s)$ (with respect to r) to get $\mathrm{res}_1(s), \mathrm{res}_2(s)$.

(2) Compute $\gcd(\mathrm{res}_1(s), \mathrm{res}_2(s))$. If degree is 0, then Jac(C) is not (N,N)-split. Otherwise, Jac(C) is (N,N)-split and s_0 is a root of the GCD. Then solve for r_0.

In fact, we obtain a more efficient method by precomputing the resultants generically.

We choose the detection set $\mathcal{D} \subseteq \{2, 3, \ldots, 11\}$ to minimise

$$\frac{\text{cost of computing Richelot isogeny from } A_c + \text{cost of (1) \& (2)}}{1 + \text{nodes revealed by detection}}.$$

Results

We implemented and optimised walks in $\Gamma_g(2; p)$ for the first step of attack in dimension $g = 1, 2$ with *and* without detection in $\Gamma_g(N; p)$.

Results

We implemented and optimised walks in $\Gamma_g(2; p)$ for the first step of attack in dimension $g = 1, 2$ with *and* without detection in $\Gamma_g(N; p)$. We ran these for different primes until reaching 10^8 $\mathbb{F}_p$ multiplications, counting number of nodes revealed and $\mathbb{F}_p$ multiplications per node revealed.

Results

We implemented and optimised walks in $\Gamma_g(2;p)$ for the first step of attack in dimension $g = 1, 2$ with *and* without detection in $\Gamma_g(N;p)$. We ran these for different primes until reaching 10^8 $\mathbb{F}_p$ multiplications, counting number of nodes revealed and $\mathbb{F}_p$ multiplications per node revealed.

		Walks in $\Gamma_g(2;p)$ without additional detection		**Walks in $\Gamma_g(2;p)$ with additional detection in $\Gamma_g(N;p)$**			
$\log p$	**Dimension** g	**nodes per 10^8 muls**	**$\mathbb{F}_p$ muls per node**	**Detection Set** $N \in \mathcal{D}$	**nodes per 10^8 muls**	**$\mathbb{F}_p$ muls per node**	**Improvement Factor**
100	1	443,951	223	$\{3,5,7\}$	2,165,681	46	4.9×
	2	85,034	1176	$\{2,3\}$	2,076,517	48	24.5×
250	1	191,115	526	$\{3,5,7,9,11,13\}$	1,607,145	62	8.4×
	2	34,083	2934	$\{4,6\}$	1,771,608	56	52.4×
500	1	97,510	1032	$\{3,5,7,9,11,13,17\}$	1,260,243	79	12.9×
	2	20,239	4941	$\{4,6\}$	1,667,360	60	82.4×
800	1	62,191	1609	$\{3,4,5,6,7,8,9,11,13,17,19\}$	1,096,056	86	17.6×
	2	13,228	7560	$\{4,6\}$	1,548,504	65	116.3×

Results

We implemented and optimised walks in $\Gamma_g(2; p)$ for the first step of attack in dimension $g = 1, 2$ with *and* without detection in $\Gamma_g(N; p)$. We ran these for different primes until reaching 10^8 $\mathbb{F}_p$ multiplications, counting number of nodes revealed and $\mathbb{F}_p$ multiplications per node revealed.

		Walks in $\Gamma_g(2; p)$ without additional detection		**Walks in $\Gamma_g(2; p)$ with additional detection in $\Gamma_g(N; p)$**			
$\log p$	**Dimension** g	**nodes per 10^8 muls**	**$\mathbb{F}_p$ muls per node**	**Detection Set** $N \in \mathcal{D}$	**nodes per 10^8 muls**	**$\mathbb{F}_p$ muls per node**	**Improvement Factor**
100	1	443,951	223	$\{3, 5, 7\}$	2,165,681	46	4.9×
	2	85,034	1176	$\{2, 3\}$	2,076,517	48	24.5×
250	1	191,115	526	$\{3, 5, 7, 9, 11, 13\}$	1,607,145	62	8.4×
	2	34,083	2934	$\{4, 6\}$	1,771,608	56	52.4×
500	1	97,510	1032	$\{3, 5, 7, 9, 11, 13, 17\}$	1,260,243	79	12.9×
	2	20,239	4941	$\{4, 6\}$	1,667,360	60	82.4×
800	1	62,191	1609	$\{3, 4, 5, 6, 7, 8, 9, 11, 13, 17, 19\}$	1,096,056	86	17.6×
	2	13,228	7560	$\{4, 6\}$	1,548,504	65	116.3×

Dimension One – SuperSolver: eprint 2021/1488
Dimension Two – SplitSearcher (*UltraSolver?*): coming soon

Results

We implemented and optimised walks in $\Gamma_g(2;p)$ for the first step of attack in dimension $g = 1, 2$ with *and* without detection in $\Gamma_g(N;p)$. We ran these for different primes until reaching 10^8 $\mathbb{F}_p$ multiplications, counting number of nodes revealed and $\mathbb{F}_p$ multiplications per node revealed.

		Walks in $\Gamma_g(2;p)$ without additional detection		**Walks in $\Gamma_g(2;p)$ with additional detection in $\Gamma_g(N;p)$**			
$\log p$	**Dimension** g	**nodes per 10^8 muls**	**$\mathbb{F}_p$ muls per node**	**Detection Set** $N \in \mathcal{D}$	**nodes per 10^8 muls**	**$\mathbb{F}_p$ muls per node**	**Improvement Factor**
100	1	443,951	223	$\{3,5,7\}$	2,165,681	46	4.9×
	2	85,034	1176	$\{2,3\}$	2,076,517	48	24.5×
250	1	191,115	526	$\{3,5,7,9,11,13\}$	1,607,145	62	8.4×
	2	34,083	2934	$\{4,6\}$	1,771,608	56	52.4×
500	1	97,510	1032	$\{3,5,7,9,11,13,17\}$	1,260,243	79	12.9×
	2	20,239	4941	$\{4,6\}$	1,667,360	60	82.4×
800	1	62,191	1609	$\{3,4,5,6,7,8,9,11,13,17,19\}$	1,096,056	86	17.6×
	2	13,228	7560	$\{4,6\}$	1,548,504	65	116.3×

Dimension One – SuperSolver: eprint 2021/1488
Dimension Two – SplitSearcher (*UltraSolver?*): coming soon
Any questions?