Fixed-Parameter Tractability of Maximum Colored Path and Beyond

Tuukka Korhonen

UNIVERSITY OF BERGEN

joint work with
Fedor V. Fomin, Petr A. Golovach, Kirill Simonov[1], and Giannos Stamoulis[2]

[1]TU Wien [2] LIRMM, Universite de Montpellier, CNRS

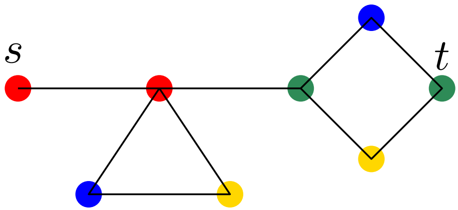
Helsinki CS Theory Seminar

10 August 2022

Maximum Colored Path

MAXIMUM COLORED *s*, *t*-PATH

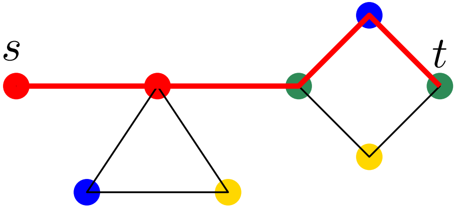
Input:	Vertex-colored undirected graph, vertices *s* and *t*, and an integer *k*.
Task:	Find an *s*, *t*-path containing at least *k* different colors.



Maximum Colored Path

MAXIMUM COLORED s, t-PATH

Input:	Vertex-colored undirected graph, vertices s and t, and an integer k.
Task:	Find an s, t-path containing at least k different colors.

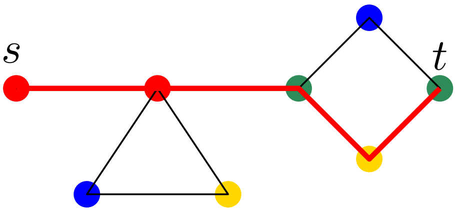


Here, such path exists when $k \leq 3$

Maximum Colored Path

MAXIMUM COLORED s, t-PATH

Input:	Vertex-colored undirected graph, vertices s and t, and an integer k.
Task:	Find an s, t-path containing at least k different colors.

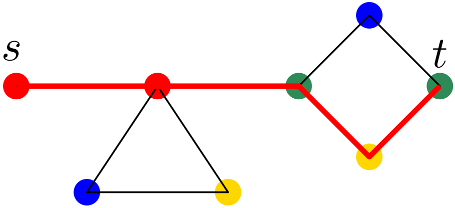


Here, such path exists when $k \leq 3$

Maximum Colored Path

Maximum Colored s, t-Path

Input:	Vertex-colored undirected graph, vertices s and t, and an integer k.
Task:	Find an s, t-path containing at least k different colors.



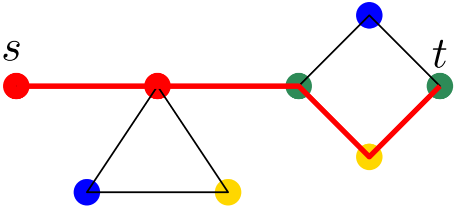
Here, such path exists when $k \leq 3$

- A path does not contain repeated vertices

Maximum Colored Path

MAXIMUM COLORED s, t-PATH

Input:	Vertex-colored undirected graph, vertices s and t, and an integer k.
Task:	Find an s, t-path containing at least k different colors.



Here, such path exists when $k \leq 3$

- A path does not contain repeated vertices
- A color may repeat multiple times in the path, and it can contain more than k colors

Fixed-parameter tractability of Maximum Colored Path

MAXIMUM COLORED s,t-PATH

Input: Vertex-colored undirected graph, vertices s and t, and an integer k.

Task: Find an s,t-path containing at least k different colors.

Theorem

There is a $2^k n^{\mathcal{O}(1)}$ time randomized algorithm for maximum colored s,t-path. Moreover, the algorithm returns the shortest solution.

Fixed-parameter tractability of Maximum Colored Path

MAXIMUM COLORED s, t-PATH

Input: Vertex-colored undirected graph, vertices s and t, and an integer k.

Task: Find an s, t-path containing at least k different colors.

Theorem

There is a $2^k n^{\mathcal{O}(1)}$ time randomized algorithm for maximum colored s, t-path. Moreover, the algorithm returns the shortest solution.

("and beyond" will come later)

Fixed-parameter tractability of Maximum Colored Path

MAXIMUM COLORED s, t-PATH

Input: Vertex-colored undirected graph, vertices s and t, and an integer k.
Task: Find an s, t-path containing at least k different colors.

Theorem

There is a $2^k n^{\mathcal{O}(1)}$ time randomized algorithm for maximum colored s, t-path. Moreover, the algorithm returns the shortest solution.

("and beyond" will come later)

- This is the first FPT algorithm (time complexity $f(k) \cdot n^{\mathcal{O}(1)}$) for this problem

Fixed-parameter tractability of Maximum Colored Path

MAXIMUM COLORED s,t-PATH

Input: Vertex-colored undirected graph, vertices s and t, and an integer k.

Task: Find an s,t-path containing at least k different colors.

Theorem

There is a $2^k n^{\mathcal{O}(1)}$ time randomized algorithm for maximum colored s,t-path. Moreover, the algorithm returns the shortest solution.

("and beyond" will come later)

- This is the first FPT algorithm (time complexity $f(k) \cdot n^{\mathcal{O}(1)}$) for this problem
- Assuming set cover conjecture, no $(2-\varepsilon)^k n^{\mathcal{O}(1)}$ time algorithm for any $\varepsilon > 0$

Fixed-parameter tractability of Maximum Colored Path

MAXIMUM COLORED s, t-PATH

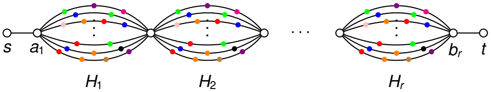
Input:	Vertex-colored undirected graph, vertices s and t, and an integer k.
Task:	Find an s, t-path containing at least k different colors.

Theorem

There is a $2^k n^{\mathcal{O}(1)}$ time randomized algorithm for maximum colored s, t-path. Moreover, the algorithm returns the shortest solution.

(“and beyond” will come later)

- This is the first FPT algorithm (time complexity $f(k) \cdot n^{\mathcal{O}(1)}$) for this problem
- Assuming set cover conjecture, no $(2 - \varepsilon)^k n^{\mathcal{O}(1)}$ time algorithm for any $\varepsilon > 0$



Fixed-parameter tractability of Maximum Colored Path

MAXIMUM COLORED s, t-PATH

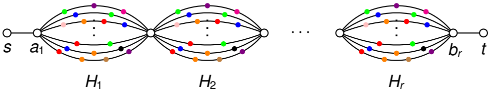
Input:	Vertex-colored undirected graph, vertices s and t, and an integer k.
Task:	Find an s, t-path containing at least k different colors.

Theorem

There is a $2^k n^{\mathcal{O}(1)}$ time randomized algorithm for maximum colored s, t-path. Moreover, the algorithm returns the shortest solution.

(“and beyond” will come later)

- This is the first FPT algorithm (time complexity $f(k) \cdot n^{\mathcal{O}(1)}$) for this problem
- Assuming set cover conjecture, no $(2 - \varepsilon)^k n^{\mathcal{O}(1)}$ time algorithm for any $\varepsilon > 0$



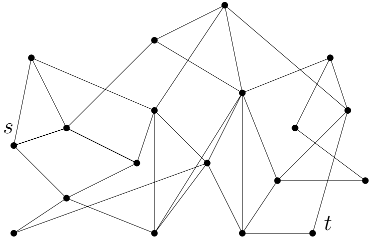
- NP-hard for directed graphs already when $k = 2$

Application: Longest path

LONGEST s, t-PATH

Input: Undirected graph, vertices s and t, and an integer k.

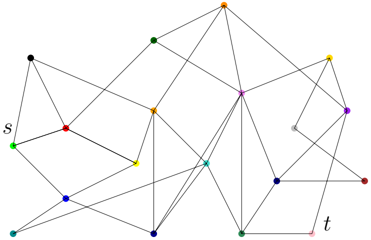
Task: Find an s, t-path of length at least k.



Application: Longest path

LONGEST s, t-PATH

Input:	Undirected graph, vertices s and t, and an integer k.
Task:	Find an s, t-path of length at least k.

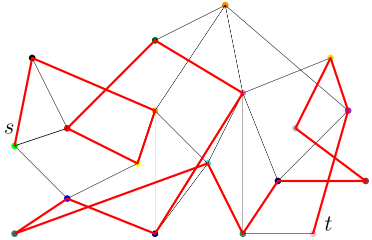


- Longest s, t-path reduces to maximum colored s, t-path by coloring all vertices with different colors

Application: Longest path

LONGEST s, t-PATH

Input:	Undirected graph, vertices s and t, and an integer k.
Task:	Find an s, t-path of length at least k.

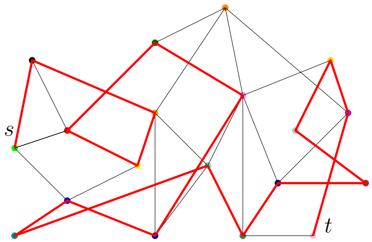


- Longest s, t-path reduces to maximum colored s, t-path by coloring all vertices with different colors

Application: Longest path

LONGEST s,t-PATH

Input:	Undirected graph, vertices s and t, and an integer k.
Task:	Find an s,t-path of length at least k.



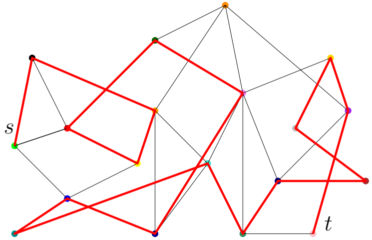
- Longest s,t-path reduces to maximum colored s,t-path by coloring all vertices with different colors

$\Rightarrow$ $2^k n^{\mathcal{O}(1)}$ time algorithm for longest s,t-path

Application: Longest path

Longest s, t-Path

Input:	Undirected graph, vertices s and t, and an integer k.
Task:	Find an s, t-path of length at least k.



- Longest s, t-path reduces to maximum colored s, t-path by coloring all vertices with different colors

$\Rightarrow$ $2^k n^{\mathcal{O}(1)}$ time algorithm for longest s, t-path

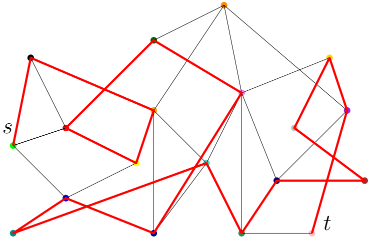
- Previous best algorithm $4.884^k n^{\mathcal{O}(1)}$ time [Fomin, Lokshtanov, Panolan, Saurabh, Zehavi'18]

Application: Longest path

Longest s, t-Path

Input: Undirected graph, vertices s and t, and an integer k.

Task: Find an s, t-path of length at least k.



- Longest s, t-path reduces to maximum colored s, t-path by coloring all vertices with different colors

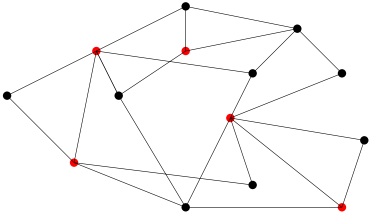
$\Rightarrow$ $2^k n^{\mathcal{O}(1)}$ time algorithm for longest s, t-path

- Previous best algorithm $4.884^k n^{\mathcal{O}(1)}$ time [Fomin, Lokshtanov, Panolan, Saurabh, Zehavi'18]
- In contrast, there is a $1.66^k n^{\mathcal{O}(1)}$ time algorithm for longest path [Björklund, Husfeldt, Kaski, Koivisto'17]

Application: T-cycle

T-CYCLE

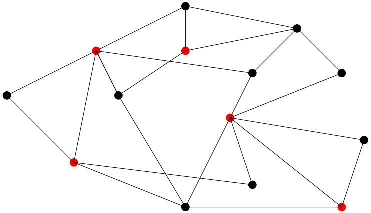
Input:	Undirected graph and a set of terminal vertices T.
Task:	Find a cycle that visits each vertex in T.



Application: T-cycle

T-CYCLE

Input:	Undirected graph and a set of terminal vertices T.
Task:	Find a cycle that visits each vertex in T.



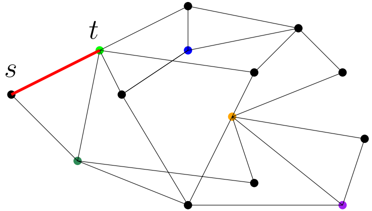
- $2^{|T|} n^{\mathcal{O}(1)}$ time algorithm for T-cycle [Björklund, Husfeldt, Taslaman'12]

Application: T-cycle

T-CYCLE

Input: Undirected graph and a set of terminal vertices T.

Task: Find a cycle that visits each vertex in T.



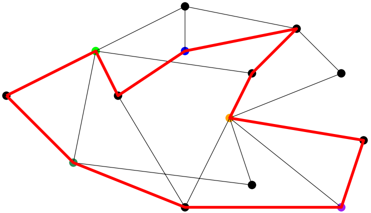
- $2^{|T|} n^{\mathcal{O}(1)}$ time algorithm for T-cycle [Björklund, Husfeldt, Taslaman'12]
- T-cycle reduces to maximum colored s, t-path by coloring the vertices in T with different colors, the rest with one color, and guessing one edge of the cycle

Application: T-cycle

T-CYCLE

Input: Undirected graph and a set of terminal vertices T.

Task: Find a cycle that visits each vertex in T.



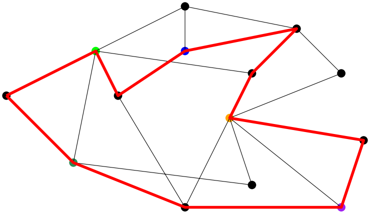
- $2^{|T|} n^{\mathcal{O}(1)}$ time algorithm for T-cycle [Björklund, Husfeldt, Taslaman'12]
- T-cycle reduces to maximum colored s, t-path by coloring the vertices in T with different colors, the rest with one color, and guessing one edge of the cycle

Application: T-cycle

T-CYCLE

Input: Undirected graph and a set of terminal vertices T.

Task: Find a cycle that visits each vertex in T.



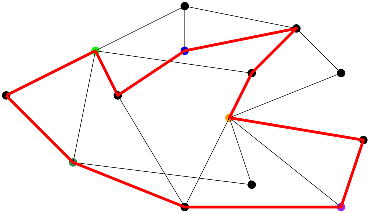
- $2^{|T|}n^{\mathcal{O}(1)}$ time algorithm for T-cycle [Björklund, Husfeldt, Taslaman'12]
- T-cycle reduces to maximum colored s,t-path by coloring the vertices in T with different colors, the rest with one color, and guessing one edge of the cycle

$\Rightarrow$ $2^{|T|}n^{\mathcal{O}(1)}$ time algorithm also via our result

Application: T-cycle

T-CYCLE

Input:	Undirected graph and a set of terminal vertices T.
Task:	Find a cycle that visits each vertex in T.



- $2^{|T|}n^{\mathcal{O}(1)}$ time algorithm for T-cycle [Björklund, Husfeldt, Taslaman'12]
- T-cycle reduces to maximum colored s,t-path by coloring the vertices in T with different colors, the rest with one color, and guessing one edge of the cycle

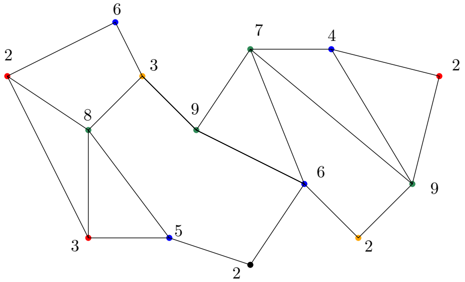
$\Rightarrow$ $2^{|T|}n^{\mathcal{O}(1)}$ time algorithm also via our result

- Allows for generalizations, e.g., arbitrarily large T

Beyond Maximum Colored Path

Input:

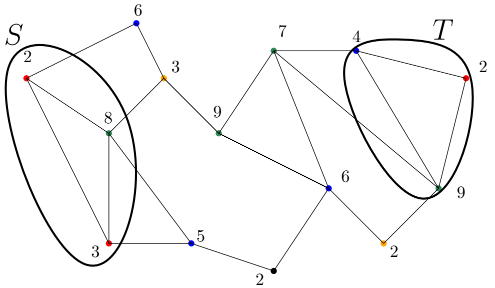
- Colored positive-integer weighted undirected graph



Beyond Maximum Colored Path

Input:

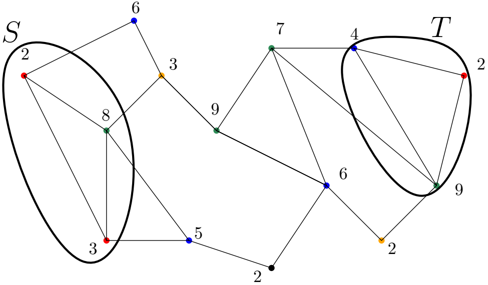
- Colored positive-integer weighted undirected graph
- Two sets of vertices S and T



Beyond Maximum Colored Path

Input:

- Colored positive-integer weighted undirected graph
- Two sets of vertices S and T
- Integers p, k, w



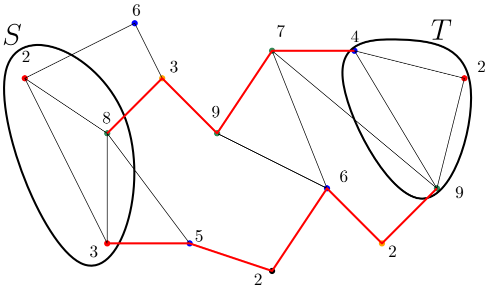
Beyond Maximum Colored Path

Input:

- Colored positive-integer weighted undirected graph
- Two sets of vertices S and T
- Integers p, k, w

Task:

- Find p vertex-disjoint paths starting in S and ending in T so that



Here, $p = 2$

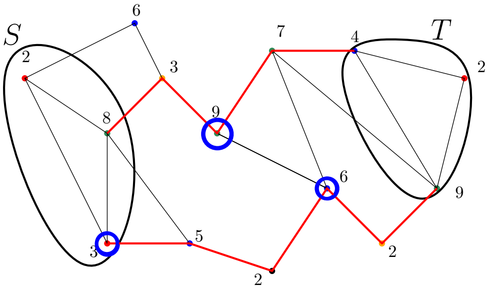
Beyond Maximum Colored Path

Input:

- Colored positive-integer weighted undirected graph
- Two sets of vertices S and T
- Integers p, k, w

Task:

- Find p vertex-disjoint paths starting in S and ending in T so that
- the vertices of the paths contain a set X of size k, total weight w, and having distinct colors



Here, $p = 2$, $k = 3$, and $w = 18$.

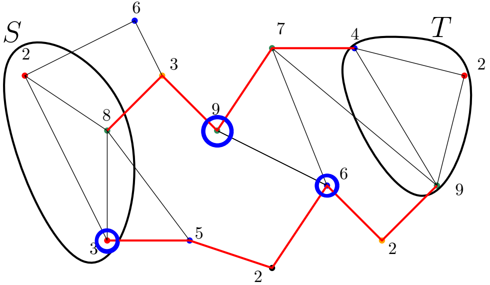
Beyond Maximum Colored Path

Input:

- Colored positive-integer weighted undirected graph
- Two sets of vertices S and T
- Integers p, k, w

Task:

- Find p vertex-disjoint paths starting in S and ending in T so that
- the vertices of the paths contain a set X of size k, total weight w, and having distinct colors
- While minimizing the total length of the paths



Here, $p = 2$, $k = 3$, and $w = 18$.

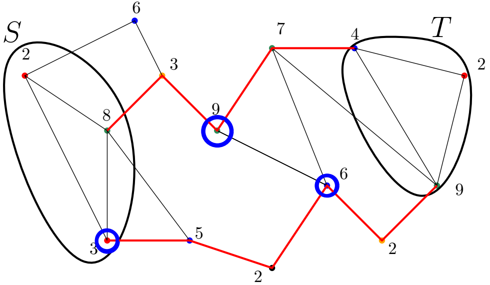
Beyond Maximum Colored Path

Input:

- Colored positive-integer weighted undirected graph
- Two sets of vertices S and T
- Integers p, k, w

Task:

- Find p vertex-disjoint paths starting in S and ending in T so that
- the vertices of the paths contain a set X of size k, total weight w, and having distinct colors
- While minimizing the total length of the paths



Here, $p = 2$, $k = 3$, and $w = 18$.

Main theorem: Randomized algorithm with time complexity $2^{k+p}n^{\mathcal{O}(1)}w$.

More applications

- Longest T-cycle in time $2^{\max(|T|,k)} n^{\mathcal{O}(1)}$

More applications

- Longest T-cycle in time $2^{\max(|T|,k)} n^{\mathcal{O}(1)}$
- Bank robber's path: Equipment for robbing k banks, know profit from each, maximize the profit while minimizing the length of the path in time $2^k n^{\mathcal{O}(1)}$

More applications

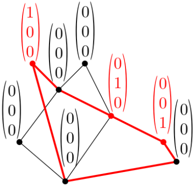
- Longest T-cycle in time $2^{\max(|T|,k)} n^{\mathcal{O}(1)}$
- Bank robber's path: Equipment for robbing k banks, know profit from each, maximize the profit while minimizing the length of the path in time $2^k n^{\mathcal{O}(1)}$
  - and generalization to p robbers in time $2^{k+p} n^{\mathcal{O}(1)}$

More applications

- Longest T-cycle in time $2^{\max(|T|,k)} n^{\mathcal{O}(1)}$
- Bank robber's path: Equipment for robbing k banks, know profit from each, maximize the profit while minimizing the length of the path in time $2^k n^{\mathcal{O}(1)}$
  - and generalization to p robbers in time $2^{k+p} n^{\mathcal{O}(1)}$
- Vehicle routing: k items, p vehicles, customers bid for items, find shortest vertex-disjoint routing to deliver items to maximize profit in time $2^{p+k} n^{\mathcal{O}(1)}$

More applications

- Longest T-cycle in time $2^{\max(|T|,k)} n^{\mathcal{O}(1)}$
- Bank robber's path: Equipment for robbing k banks, know profit from each, maximize the profit while minimizing the length of the path in time $2^k n^{\mathcal{O}(1)}$
  - and generalization to p robbers in time $2^{k+p} n^{\mathcal{O}(1)}$
- Vehicle routing: k items, p vehicles, customers bid for items, find shortest vertex-disjoint routing to deliver items to maximize profit in time $2^{p+k} n^{\mathcal{O}(1)}$
- Generalization from colored graphs to (graph, matroid) pairs represented over a finite field of order q with a $2^{\mathcal{O}(k^2 \log(q+p))}$ factor overhead



The algorithm

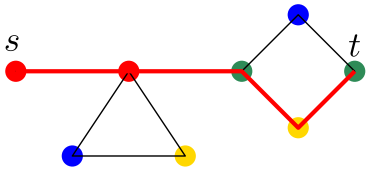
Outline

In this talk, we focus on:

MAXIMUM COLORED s, t-PATH

Input: Vertex-colored undirected graph, vertices s and t, and an integer k.

Task: Find a k-colored s, t-path. (s, t-path with at least k different colors)

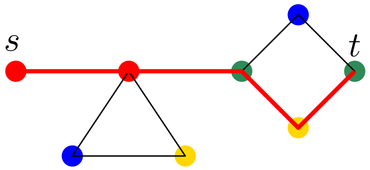


Outline

In this talk, we focus on:

MAXIMUM COLORED s, t-PATH

Input:	Vertex-colored undirected graph, vertices s and t, and an integer k.
Task:	Find a k-colored s, t-path. (s, t-path with at least k different colors)



- Algorithm based on algebraic approach, extending ideas that were developed by [Björklund, Husfeldt, Taslaman'12], [Björklund'14], [Björklund, Husfeldt, Kaski, Koivisto'17] for T-cycle, hamiltonicity, and k-path

Algebraic approach

General idea:

Algebraic approach

General idea:

- Design a multivariate polynomial $p(x_1, \ldots, x_m)$ of total degree $\leq 4n$ over $\mathrm{GF}(2^{3+\lceil \log n \rceil})$ (finite field of order $\geq 8n$ and characteristic 2) so that

Algebraic approach

General idea:

- Design a multivariate polynomial $p(x_1, \ldots, x_m)$ of total degree $\leq 4n$ over $\mathrm{GF}(2^{3+\lceil \log n \rceil})$ (finite field of order $\geq 8n$ and characteristic 2) so that
  1. no-instance $\rightarrow p(x_1, \ldots, x_m)$ is identically zero polynomial

Algebraic approach

General idea:

- Design a multivariate polynomial $p(x_1, \ldots, x_m)$ of total degree $\leq 4n$ over $\mathrm{GF}(2^{3+\lceil \log n \rceil})$ (finite field of order $\geq 8n$ and characteristic 2) so that
  1. no-instance $\rightarrow p(x_1, \ldots, x_m)$ is identically zero polynomial
  2. yes-instance $\rightarrow p(x_1, \ldots, x_m)$ is a non-zero polynomial

Algebraic approach

General idea:

- Design a multivariate polynomial $p(x_1, \ldots, x_m)$ of total degree $\leq 4n$ over $\mathrm{GF}(2^{3+\lceil \log n \rceil})$ (finite field of order $\geq 8n$ and characteristic 2) so that
  1. no-instance $\rightarrow p(x_1, \ldots, x_m)$ is identically zero polynomial
  2. yes-instance $\rightarrow p(x_1, \ldots, x_m)$ is a non-zero polynomial
  3. Given $x_1, \ldots, x_m$, $p(x_1, \ldots, x_m)$ can be evaluated in $2^k n^{\mathcal{O}(1)}$ time

Algebraic approach

General idea:

- Design a multivariate polynomial $p(x_1, \ldots, x_m)$ of total degree $\leq 4n$ over $\mathrm{GF}(2^{3+\lceil \log n \rceil})$ (finite field of order $\geq 8n$ and characteristic 2) so that
  1. no-instance $\rightarrow p(x_1, \ldots, x_m)$ is identically zero polynomial
  2. yes-instance $\rightarrow p(x_1, \ldots, x_m)$ is a non-zero polynomial
  3. Given $x_1, \ldots, x_m$, $p(x_1, \ldots, x_m)$ can be evaluated in $2^k n^{\mathcal{O}(1)}$ time

- By DeMillo–Lipton–Schwartz–Zippel lemma, the problem is then solved in $2^k n^{\mathcal{O}(1)}$ time by evaluating $p(x_1, \ldots, x_m)$ for random $x_1, \ldots, x_m$.

Algebraic approach

General idea:

- Design a multivariate polynomial $p(x_1, \ldots, x_m)$ of total degree $\leq 4n$ over $\mathrm{GF}(2^{3+\lceil \log n \rceil})$ (finite field of order $\geq 8n$ and characteristic 2) so that
  1. no-instance $\rightarrow p(x_1, \ldots, x_m)$ is identically zero polynomial
  2. yes-instance $\rightarrow p(x_1, \ldots, x_m)$ is a non-zero polynomial
  3. Given $x_1, \ldots, x_m$, $p(x_1, \ldots, x_m)$ can be evaluated in $2^k n^{\mathcal{O}(1)}$ time
- By DeMillo–Lipton–Schwartz–Zippel lemma, the problem is then solved in $2^k n^{\mathcal{O}(1)}$ time by evaluating $p(x_1, \ldots, x_m)$ for random $x_1, \ldots, x_m$.
- Characteristic 2?
  - $x + x = 0$ for any x

Design of the polynomial

Idea: Design a polynomial where monomials correspond to *labeled walks*

Design of the polynomial

Idea: Design a polynomial where monomials correspond to *labeled walks*

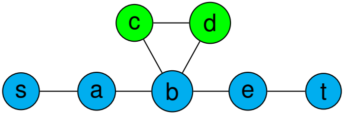
- Evaluating a polynomial over walks is easier than over paths

Design of the polynomial

Idea: Design a polynomial where monomials correspond to *labeled walks*

- Evaluating a polynomial over walks is easier than over paths
- Argue that monomials corresponding to non-path walks and paths with less than k colors cancel out (by $x + x = 0$), and only the contribution of k-colored paths remains

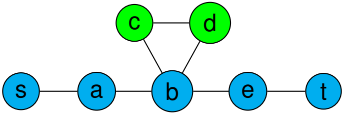
Labeled walks



$k = 2$

- (s, t)-walk of length ℓ is a sequence $s = v_1 \dots v_\ell = t$ of ℓ adjacent vertices

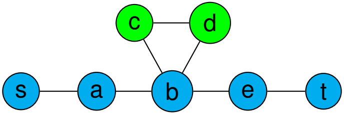
Labeled walks



$k = 2$

- (s, t)-walk of length ℓ is a sequence $s = v_1 \dots v_\ell = t$ of ℓ adjacent vertices
  - For example *sabcdbet* is an (s, t)-walk of length 8.

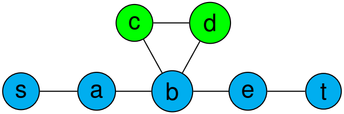
Labeled walks



$k = 2$

- (s, t)-walk of length ℓ is a sequence $s = v_1 \ldots v_\ell = t$ of ℓ adjacent vertices
  - For example *sabcdbet* is an (s, t)-walk of length 8.
- Labeled (s, t)-walk is an (s, t)-walk where k indices are labeled with k different labels from $[1, k]$

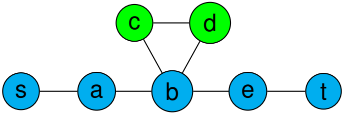
Labeled walks



$k = 2$

- (s, t)-walk of length ℓ is a sequence $s = v_1 \dots v_\ell = t$ of ℓ adjacent vertices
  - For example $sabcdbet$ is an (s, t)-walk of length 8.
- Labeled (s, t)-walk is an (s, t)-walk where k indices are labeled with k different labels from $[1, k]$
  - For example $sa\overset{1}{b}c\overset{2}{d}bet$ is a labeled (s, t)-walk ($\overset{1}{b}$ labeled with label 1 and $\overset{2}{d}$ labeled with label 2)

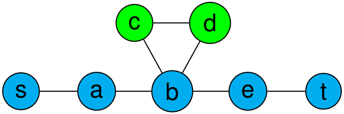
Labeled walks



$k = 2$

- (s, t)-walk of length ℓ is a sequence $s = v_1 \dots v_\ell = t$ of ℓ adjacent vertices
  - For example *sabcdbet* is an (s, t)-walk of length 8.
- Labeled (s, t)-walk is an (s, t)-walk where k indices are labeled with k different labels from $[1, k]$
  - For example $sa\overset{1}{b}c\overset{2}{d}bet$ is a labeled (s, t)-walk ($\overset{1}{b}$ labeled with label 1 and $\overset{2}{d}$ labeled with label 2)
  - Intention of labels: Indicate k vertices with different colors

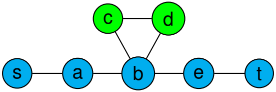
Labeled walks



$k = 2$

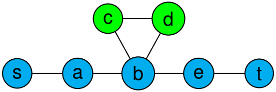
- (s, t)-walk of length ℓ is a sequence $s = v_1 \dots v_\ell = t$ of ℓ adjacent vertices
  - For example *sabcdbet* is an (s, t)-walk of length 8.
- Labeled (s, t)-walk is an (s, t)-walk where k indices are labeled with k different labels from $[1, k]$
  - For example $sa\overset{1}{b}c\overset{2}{d}bet$ is a labeled (s, t)-walk ($\overset{1}{b}$ labeled with label 1 and $\overset{2}{d}$ labeled with label 2)
  - Intention of labels: Indicate k vertices with different colors
- A labeled (s, t)-walk is *labeled-digon-free* if it has no subwalks of form $a\overset{1}{b}a$

Polynomial over labeled walks



$k = 2$

Polynomial over labeled walks

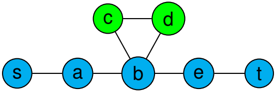


$k = 2$

Definition:

- For each edge uv associate variable $f_e(uv)$
- For each vertex w associate variable $f_v(w)$
- For each color-label pair (x, y) associate variable $f_c(x, y)$

Polynomial over labeled walks



$k = 2$

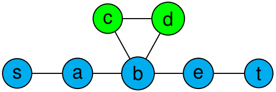
Definition:

- For each edge uv associate variable $f_e(uv)$
- For each vertex w associate variable $f_v(w)$
- For each color-label pair (x, y) associate variable $f_c(x, y)$

For a labeled walk W, associate monomial $f(W)$ that is product of edge variables, vertex variables of labeled vertices, and color-label pair variables corresponding to labeled vertices

$$f(sa\overset{1}{b}c\overset{2}{d}bet) = f_e(sa)f_e(ab)f_e(bc)f_e(cd)f_e(db)f_e(be)f_e(et)f_v(b)f_v(d)f_c(\bullet, 1)f_c(\bullet, 2)$$

Polynomial over labeled walks



$k = 2$

Definition:

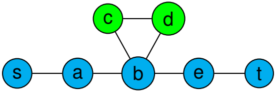
- For each edge uv associate variable $f_e(uv)$
- For each vertex w associate variable $f_v(w)$
- For each color-label pair (x, y) associate variable $f_c(x, y)$

For a labeled walk W, associate monomial $f(W)$ that is product of edge variables, vertex variables of labeled vertices, and color-label pair variables corresponding to labeled vertices

$$f(sa\overset{1}{b}c\overset{2}{d}bet) = f_e(sa)f_e(ab)f_e(bc)f_e(cd)f_e(db)f_e(be)f_e(et)f_v(b)f_v(d)f_c(\bullet, 1)f_c(\bullet, 2)$$

For integer ℓ, let $\mathcal{C}_\ell$ be the family of labeled-digon-free labeled (s, t)-walks of length ℓ

Polynomial over labeled walks



$k = 2$

Definition:

- For each edge uv associate variable $f_e(uv)$
- For each vertex w associate variable $f_v(w)$
- For each color-label pair (x, y) associate variable $f_c(x, y)$

For a labeled walk W, associate monomial $f(W)$ that is product of edge variables, vertex variables of labeled vertices, and color-label pair variables corresponding to labeled vertices

$$f(sa\overset{1}{b}c\overset{2}{d}bet) = f_e(sa)f_e(ab)f_e(bc)f_e(cd)f_e(db)f_e(be)f_e(et)f_v(b)f_v(d)f_c(\bullet, 1)f_c(\bullet, 2)$$

For integer ℓ, let $\mathcal{C}_\ell$ be the family of labeled-digon-free labeled (s, t)-walks of length ℓ

- Define $f(\mathcal{C}_\ell) = \sum_{W \in \mathcal{C}_\ell} f(W)$

Algebraic approach

Now we claim that:

Algebraic approach

Now we claim that:

1. Exists a k-colored (s, t)-path of length ℓ $\quad \Rightarrow \quad$ $f(\mathcal{C}_\ell)$ non-zero

Algebraic approach

Now we claim that:

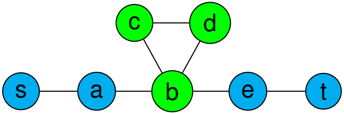
1. Exists a k-colored (s, t)-path of length ℓ $\quad \Rightarrow \quad$ $f(\mathcal{C}_\ell)$ non-zero
2. No k-colored (s, t)-path of length $\leq \ell$ $\quad \Rightarrow \quad$ $f(\mathcal{C}_\ell)$ identically zero

Algebraic approach

Now we claim that:

1. Exists a k-colored (s,t)-path of length ℓ $\Rightarrow$ $f(\mathcal{C}_\ell)$ non-zero
2. No k-colored (s,t)-path of length $\leq \ell$ $\Rightarrow$ $f(\mathcal{C}_\ell)$ identically zero
3. $f(\mathcal{C}_\ell)$ can be evaluated in time $2^k n^{\mathcal{O}(1)}$

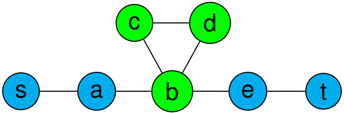
Yes-instances



$k = 2$

(1) Exists a k-colored (s, t)-path of length ℓ $\quad\Rightarrow\quad$ $f(\mathcal{C}_\ell)$ non-zero

Yes-instances

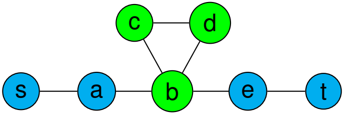


$k = 2$

(1) Exists a k-colored (s, t)-path of length ℓ $\Rightarrow$ $f(\mathcal{C}_\ell)$ non-zero

- The labeled walk $s\overset{1}{a}\overset{2}{b}et \in \mathcal{C}_5$ contributes the monomial $f_e(sa)f_e(ab)f_e(be)f_e(et)f_v(a)f_v(b)f_c(\bullet, 1)f_c(\bullet, 2)$

Yes-instances

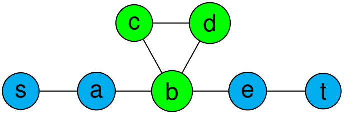


$k = 2$

(1) Exists a k-colored (s,t)-path of length ℓ $\quad\Rightarrow\quad$ $f(\mathcal{C}_\ell)$ non-zero

- The labeled walk $s\overset{1}{a}\overset{2}{b}et \in \mathcal{C}_5$ contributes the monomial $f_e(sa)f_e(ab)f_e(be)f_e(et)f_v(a)f_v(b)f_c(\bullet, 1)f_c(\bullet, 2)$
- No other labeled walk in $\mathcal{C}_5$ contributes the same monomial

Yes-instances



$k = 2$

(1) Exists a k-colored (s,t)-path of length ℓ $\quad\Rightarrow\quad$ $f(\mathcal{C}_\ell)$ non-zero

- The labeled walk $s\overset{1}{a}\overset{2}{b}et \in \mathcal{C}_5$ contributes the monomial $f_e(sa)f_e(ab)f_e(be)f_e(et)f_v(a)f_v(b)f_c(\bullet, 1)f_c(\bullet, 2)$
- No other labeled walk in $\mathcal{C}_5$ contributes the same monomial

$\Rightarrow$ $f(\mathcal{C}_5)$ is non-zero

No-instances

(2) No k-colored (s, t)-path of length $\leq \ell \quad \Rightarrow \quad f(\mathcal{C}_\ell)$ identically zero

No-instances

(2) No k-colored (s, t)-path of length $\leq \ell \quad \Rightarrow \quad f(\mathcal{C}_\ell)$ identically zero

Goal: Design a function $\phi : \mathcal{C}_\ell \to \mathcal{C}_\ell$ so that for all $W \in \mathcal{C}_\ell$

- $f(W) = f(\phi(W))$
- $\phi(W) \neq W$
- $\phi(\phi(W)) = W$

No-instances

(2) No k-colored (s,t)-path of length $\leq \ell \quad \Rightarrow \quad f(\mathcal{C}_\ell)$ identically zero

Goal: Design a function $\phi : \mathcal{C}_\ell \to \mathcal{C}_\ell$ so that for all $W \in \mathcal{C}_\ell$

- $f(W) = f(\phi(W))$
- $\phi(W) \neq W$
- $\phi(\phi(W)) = W$

$\Rightarrow$ Labeled walks in $\mathcal{C}_\ell$ can be paired as $\{W, \phi(W)\}$, implying everything cancels out over fields of characteristic 2

No-instances

(2) No k-colored (s, t)-path of length $\leq \ell \quad \Rightarrow \quad f(\mathcal{C}_\ell)$ identically zero

Goal: Design a function $\phi : \mathcal{C}_\ell \to \mathcal{C}_\ell$ so that for all $W \in \mathcal{C}_\ell$

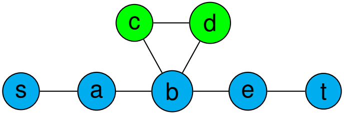
- $f(W) = f(\phi(W))$
- $\phi(W) \neq W$
- $\phi(\phi(W)) = W$

$\Rightarrow$ Labeled walks in $\mathcal{C}_\ell$ can be paired as $\{W, \phi(W)\}$, implying everything cancels out over fields of characteristic 2

Three different cancellation arguments as building blocks for ϕ:

- Bijective-labeling based cancellation (from [Björklund'14], [Björklund, Husfeldt, Kaski, Koivisto'17])
- Cycle-reversal based cancellation (from [Björklund, Husfeldt, Taslaman'12])
- Label-swap based cancellation

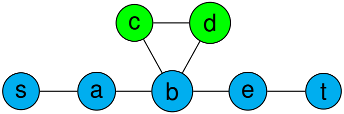
Bijective-labeling based cancellation



$k = 2$

Consider labeled walks where two vertices of the same color are labeled

Bijective-labeling based cancellation

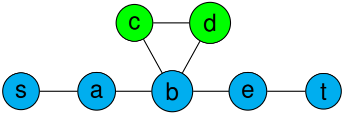


$k = 2$

Consider labeled walks where two vertices of the same color are labeled

- $s\overset{1}{a}b\overset{2}{e}t \in \mathcal{C}_5$ contributes the monomial $f_e(sa)f_e(ab)f_e(be)f_e(et)f_v(a)f_v(e)f_c(\bullet, 1)f_c(\bullet, 2)$

Bijective-labeling based cancellation

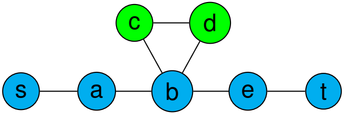


$k = 2$

Consider labeled walks where two vertices of the same color are labeled

- $s\overset{1}{a}b\overset{2}{e}t \in \mathcal{C}_5$ contributes the monomial
  $f_e(sa)f_e(ab)f_e(be)f_e(et)f_v(a)f_v(e)f_c(\bullet, 1)f_c(\bullet, 2)$
- $s\overset{2}{a}b\overset{1}{e}t \in \mathcal{C}_5$ is a different labeled walk, but contributes the same monomial

Bijective-labeling based cancellation

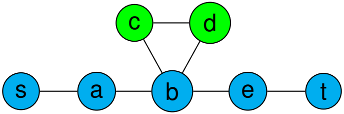


$k = 2$

Consider labeled walks where two vertices of the same color are labeled

- $s\overset{1}{a}b\overset{2}{e}t \in \mathcal{C}_5$ contributes the monomial $f_e(sa)f_e(ab)f_e(be)f_e(et)f_v(a)f_v(e)f_c(\bullet, 1)f_c(\bullet, 2)$
- $s\overset{2}{a}b\overset{1}{e}t \in \mathcal{C}_5$ is a different labeled walk, but contributes the same monomial
- $\phi(s\overset{1}{a}b\overset{2}{e}t) = s\overset{2}{a}b\overset{1}{e}t$ and $\phi(s\overset{2}{a}b\overset{1}{e}t) = s\overset{1}{a}b\overset{2}{e}t$

Bijective-labeling based cancellation

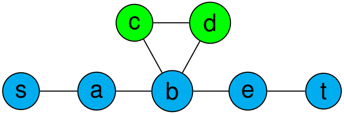


$k = 2$

Consider labeled walks where two vertices of the same color are labeled

- $s\overset{1}{a}b\overset{2}{e}t \in \mathcal{C}_5$ contributes the monomial $f_e(sa)f_e(ab)f_e(be)f_e(et)f_v(a)f_v(e)f_c(\bullet, 1)f_c(\bullet, 2)$
- $s\overset{2}{a}b\overset{1}{e}t \in \mathcal{C}_5$ is a different labeled walk, but contributes the same monomial
- $\phi(s\overset{1}{a}b\overset{2}{e}t) = s\overset{2}{a}b\overset{1}{e}t$ and $\phi(s\overset{2}{a}b\overset{1}{e}t) = s\overset{1}{a}b\overset{2}{e}t$
- Now we can work with family $\mathcal{C}_\ell^* \subseteq \mathcal{C}_\ell$ containing labeled walks where all labeled vertices have different colors

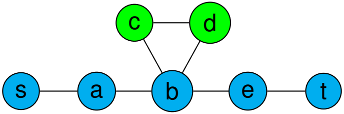
Cycle-reversal based cancellation



$k = 2$

- Labeled walk $s\overset{1}{a}b\overset{2}{c}dbet \in \mathcal{C}_7^*$ contributes monomial $f_e(sa)f_e(ab)f_e(bc)f_e(cd)f_e(db)f_e(be)f_e(et)f_v(a)f_v(c)f_c(\bullet, 1)f_c(\bullet, 2)$

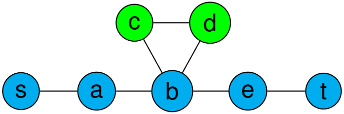
Cycle-reversal based cancellation



$k = 2$

- Labeled walk $\color{red}{s\overset{1}{a}b\overset{2}{c}dbet} \in \mathcal{C}_7^*$ contributes monomial $f_e(sa)f_e(ab)f_e(bc)f_e(cd)f_e(db)f_e(be)f_e(et)f_v(a)f_v(c)f_c(\color{blue}{\bullet}, 1)f_c(\color{green}{\bullet}, 2)$
- Reverse the cycle $\color{red}{b\overset{2}{c}db}$

Cycle-reversal based cancellation



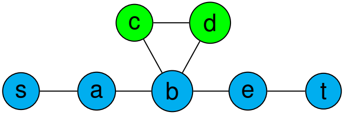
$k = 2$

- Labeled walk $s\overset{1}{a}b\overset{2}{c}dbet \in \mathcal{C}_7^*$ contributes monomial $f_e(sa)f_e(ab)f_e(bc)f_e(cd)f_e(db)f_e(be)f_e(et)f_v(a)f_v(c)f_c(\bullet, 1)f_c(\bullet, 2)$

- Reverse the cycle $b\overset{2}{c}db$

- Now $s\overset{1}{a}bd\overset{2}{c}bet \in \mathcal{C}_7^*$ is a different labeled walk but contributes the same monomial

Cycle-reversal based cancellation



$k = 2$

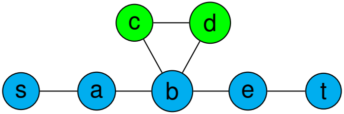
- Labeled walk $s\overset{1}{a}b\overset{2}{c}dbet \in \mathcal{C}_7^*$ contributes monomial $f_e(sa)f_e(ab)f_e(bc)f_e(cd)f_e(db)f_e(be)f_e(et)f_v(a)f_v(c)f_c(\color{blue}{\bullet}, 1)f_c(\color{green}{\bullet}, 2)$

- Reverse the cycle $b\overset{2}{c}db$

- Now $s\overset{1}{a}bd\overset{2}{c}bet \in \mathcal{C}_7^*$ is a different labeled walk but contributes the same monomial

- $\phi(s\overset{1}{a}b\overset{2}{c}dbet) = s\overset{1}{a}bd\overset{2}{c}bet$ and $\phi(s\overset{1}{a}bd\overset{2}{c}bet) = s\overset{1}{a}b\overset{2}{c}dbet$

Cycle-reversal based cancellation



$k = 2$

- Labeled walk $\color{red}{s\overset{1}{a}b\overset{2}{c}dbet} \in \mathcal{C}_7^*$ contributes monomial $f_e(sa)f_e(ab)f_e(bc)f_e(cd)f_e(db)f_e(be)f_e(et)f_v(a)f_v(c)f_c(\color{blue}{\bullet}, 1)f_c(\color{green}{\bullet}, 2)$

- Reverse the cycle $\color{red}{b\overset{2}{c}db}$

- Now $\color{red}{s\overset{1}{a}bd\overset{2}{c}bet} \in \mathcal{C}_7^*$ is a different labeled walk but contributes the same monomial

- $\phi(\color{red}{s\overset{1}{a}b\overset{2}{c}dbet}) = \color{red}{s\overset{1}{a}bd\overset{2}{c}bet}$ and $\phi(\color{red}{s\overset{1}{a}bd\overset{2}{c}bet}) = \color{red}{s\overset{1}{a}b\overset{2}{c}dbet}$

- This does **not** cleanly handle things, need more arguments and case analysis

Label-swap based cancellation

- Consider a labeled walk $sabcd\overset{1}{a}bt$ (with only one label)

Label-swap based cancellation

- Consider a labeled walk $sabcd\overset{1}{a}bt$ (with only one label)
- Reversing the cycle $abcd\overset{1}{a}$ would create $sadcb\overset{1}{a}bt$, containing labeled digon $b\overset{1}{a}b$

Label-swap based cancellation

- Consider a labeled walk $sabcd\overset{1}{a}bt$ (with only one label)
- Reversing the cycle $abcd\overset{1}{a}$ would create $sadcb\overset{1}{a}bt$, containing labeled digon $b\overset{1}{a}b$
- Label swap operation: $\phi(sabcd\overset{1}{a}bt) = s\overset{1}{a}bcdabt$ and $\phi(s\overset{1}{a}bcdabt) = sabcd\overset{1}{a}bt$

Label-swap based cancellation

- Consider a labeled walk $sabcd\overset{1}{a}bt$ (with only one label)
- Reversing the cycle $abcd\overset{1}{a}$ would create $sadcb\overset{1}{a}bt$, containing labeled digon $b\overset{1}{a}b$
- Label swap operation: $\phi(sabcd\overset{1}{a}bt) = s\overset{1}{a}bcdabt$ and $\phi(s\overset{1}{a}bcdabt) = sabcd\overset{1}{a}bt$
- The label swap operation could also create labeled digons!

Cancellation arguments

- Bad news: Not clear when to apply cycle reversal and when label swap

Cancellation arguments

- Bad news: Not clear when to apply cycle reversal and when label swap
- Good news: We managed to construct ϕ, but it is very complicated

Definition 3 (The function ϕ). *Let $\mathcal{W} = (W^1, \dots, W^p)$ be a proper barren labeled walkage of order p. For each $i \in [p]$, denote $W^i = ((v^i_1, \dots, v^i_{\ell_i}), (r^i_1, \dots, r^i_{\ell_i}))$. The value $\phi(\mathcal{W})$ is defined, in some cases recursively, by selecting the first matching case from the following list:*

A. *if the vertex v^1_1 occurs only once in $\mathcal{W}$:*

   1. *if $\ell_1 \geq 2$, then $\phi(\mathcal{W}) = W^1[1,1] \diamond \phi(W^1[2,\ell_1], W^2, \dots, W^p)$.*
   2. *otherwise (i.e., $\ell_1 = 1$), $\phi(\mathcal{W}) = W^1 \sqcup \phi(W^2, \dots, W^p)$.*

B. *if the vertex v^1_1 occurs in at least three different walks W^i:*
   There must be at least two different walks W^i that contain v^1_1 but do not contain it as labeled. Let i, j be the two smallest indices so that both W^i and W^j contain v^1_1 but do not contain it as labeled. Let a be the index of the first occurrence of v^1_1 in W^i and b be the index of the first occurrence of v^1_1 in W^j. Now, $\phi(\mathcal{W}) = \mathcal{W} \leftrightarrow^{i,j}_{a,b}$.

C. *if the vertex v^1_1 occurs only in the walk W^1:*
   By the case (A), the vertex v^1_1 occurs multiple times in W^1. Let b be the index of the last occurrence of v^1_1 in W^1 and a be the index of the second last occurrence of v^1_1 in W^1. Note that $a = 1$ if v^1_1 occurs only twice in W^1, and note also that $1 \leq a \leq b-2$.

   1. *if $r^1_1 = r^1_b = 0$:*
      (a) *if $W^1[2, b-1]$ is not a palindrome, then $\phi(\mathcal{W}) = (W^1\overleftarrow{[2, b-1]}, W^2, \dots, W^p)$.*
      (b) *otherwise, if $b < \ell_1$, then $\phi(\mathcal{W}) = W^1[1,b] \diamond \phi(W^1[b+1, \ell_1], W^2, \dots, W^p)$.*
      (c) *otherwise (i.e., $b = \ell_1$), $\phi(\mathcal{W}) = W^1 \sqcup \phi(W^2, \dots, W^p)$.*
   2. *if the index b is not a digon in W^1, then $\phi(\mathcal{W}) = \mathcal{W} \frown^{1,1}_{1,b}$.*
      Note: If neither case (1) nor (2) applies, then $r^1_1 \neq 0$.
   3. *if $W^1[2, a-1]$ is not a palindrome, then $\phi(\mathcal{W}) = (W^1\overleftarrow{[2, a-1]}, W^2, \dots, W^p)$.*
      Note: If $a = 1$, then $W^1[2, a-1]$ is the empty walk which is a palindrome.
   4. *if $v^1_{a+1} = v^1_{b-1}$:*
      (a) *if $W^1[a+1, b-1]$ is not a palindrome, then $\phi(\mathcal{W}) = (W^1\overleftarrow{[a+1, b-1]}, W^2, \dots, W^p)$.*
      (b) *otherwise, $\phi(\mathcal{W}) = W^1[1,b] \diamond \phi(W^1[b+1, \ell_1], W^2, \dots, W^p)$.*
      Note: Here $W^1[b+1, \ell_1]$ cannot be an empty walk because by case (C.2) b is a digon in W^1.

   X. *otherwise, $\phi(\mathcal{W}) = W^1[1,a] \diamond \phi(W^1[a+1, \ell_1], W^2, \dots, W^p)$.*
      Note: The case C.X will form a "common case" with the case D.X.

D. *if the vertex v^1_1 occurs in exactly two different walks:*
   Let i be the index of the another walk W^i in which v^1_1 occurs and let b be the index of the first occurrence of v^1_1 in W^i.

   1. *if $r^1_1 = r^i_b = 0$, then $\phi(\mathcal{W}) = \mathcal{W} \leftrightarrow^{1,i}_{1,b}$.*
   2. *if the index b is not a digon in W^i, then $\phi(\mathcal{W}) = \mathcal{W} \frown^{1,i}_{1,b}$.*
      Note: If neither case (1) nor (2) applies, then $r^1_1 \neq 0$.
   3. *if v^1_1 occurs at least twice in W^i, then let c be the index of its second occurrence and $\phi(\mathcal{W}) = \mathcal{W} \leftrightarrow^{1,i}_{2,c+1}$.*
      Note: It can happen that one of the suffixes in this case is empty. However, both of them cannot be empty at the same time because W^1 and W^i have different ending vertices because $\mathcal{W}$ is proper.
      Note: In the remaining cases, v^1_1 occurs exactly once in W^i, and this occurrence is a digon at index b.
      Now, let a be the index of the last occurrence of v^1_1 in W^1 (if v^1_1 occurs only once in W^1, then $a = 1$).
   4. *if $W^1[2, a-1]$ is not a palindrome, then $\phi(\mathcal{W}) = (W^1\overleftarrow{[2, a-1]}, W^2, \dots, W^p)$.*
      Note: If $a = 1$, then $W^1[2, a-1]$ is the empty walk which is a palindrome.
   5. *if $a = \ell_1$, then $\phi(\mathcal{W}) = W^1 \sqcup \phi(W^2, \dots, W^p)$.*
   6. *if $v^1_{a+1} = v^i_{b+1}$, then $\phi(\mathcal{W}) = \mathcal{W} \leftrightarrow^{1,i}_{a+1,b+1}$.*
      Note: By case (5) it holds that $a < \ell_1$ and by case (2) it holds that $b < \ell_i$.

   X. *otherwise, $\phi(\mathcal{W}) = W^1[1,a] \diamond \phi(W^1[a+1, \ell_1], W^2, \dots, W^p)$.*
      Note: The case D.X will form a "common case" with the case C.X

Cancellation arguments

- Bad news: Not clear when to apply cycle reversal and when label swap
- Good news: We managed to construct ϕ, but it is very complicated
  - Single-path version has 8 cases, multi-path version 18 cases

Definition 3 (The function ϕ). *Let $\mathcal{W} = (W^1, \dots, W^p)$ be a proper barren labeled walkage of order p. For each $i \in [p]$, denote $W^i = ((v^i_1, \dots, v^i_{\ell_i}), (r^i_1, \dots, r^i_{\ell_i}))$. The value $\phi(\mathcal{W})$ is defined, in some cases recursively, by selecting the first matching case from the following list:*

A. if the vertex v^1_1 occurs only once in $\mathcal{W}$:

1. *if $\ell_1 \geq 2$, then $\phi(\mathcal{W}) = W^1[1,1] \diamond \phi(W^1[2,\ell_1], W^2, \dots, W^p)$.*
2. *otherwise (i.e., $\ell_1 = 1$), $\phi(\mathcal{W}) = W^1 \sqcup \phi(W^2, \dots, W^p)$.*

B. if the vertex v^1_1 occurs in at least three different walks W^i:
There must be at least two different walks W^i that contain v^1_1 but do not contain it as labeled. Let i, j be the two smallest indices so that both W^i and W^j contain v^1_1 but do not contain it as labeled. Let a be the index of the first occurrence of v^1_1 in W^i and b be the index of the first occurrence of v^1_1 in W^j. Now, $\phi(\mathcal{W}) = \mathcal{W} \leftrightarrow^{i,j}_{a,b}$.

C. if the vertex v^1_1 occurs only in the walk W^1:
By the case (A), the vertex v^1_1 occurs multiple times in W^1. Let b be the index of the last occurrence of v^1_1 in W^1 and a be the index of the second last occurrence of v^1_1 in W^1. Note that $a = 1$ if v^1_1 occurs only twice in W^1, and note also that $1 \leq a \leq b - 2$.

1. *if $r^1_1 = r^1_b = 0$:*
   (a) *if $W^1[2, b-1]$ is not a palindrome, then $\phi(\mathcal{W}) = (W^1\overleftarrow{[2, b-1]}, W^2, \dots, W^p)$.*
   (b) *otherwise, if $b < \ell_1$, then $\phi(\mathcal{W}) = W^1[1, b] \diamond \phi(W^1[b+1, \ell_1], W^2, \dots, W^p)$.*
   (c) *otherwise (i.e., $b = \ell_1$), $\phi(\mathcal{W}) = W^1 \sqcup \phi(W^2, \dots, W^p)$.*
2. *if the index b is not a digon in W^1, then $\phi(\mathcal{W}) = \mathcal{W} \frown^{1,1}_{1,b}$.*
   Note: If neither case (1) nor (2) applies, then $r^1_1 \neq 0$.
3. *if $W^1[2, a-1]$ is not a palindrome, then $\phi(\mathcal{W}) = (W^1\overleftarrow{[2, a-1]}, W^2, \dots, W^p)$.*
   Note: If $a = 1$, then $W^1[2, a-1]$ is the empty walk which is a palindrome.
4. *if $v^1_{a+1} = v^1_{b-1}$:*
   (a) *if $W^1[a+1, b-1]$ is not a palindrome, then $\phi(\mathcal{W}) = (W^1\overleftarrow{[a+1, b-1]}, W^2, \dots, W^p)$.*
   (b) *otherwise, $\phi(\mathcal{W}) = W^1[1, b] \diamond \phi(W^1[b+1, \ell_1], W^2, \dots, W^p)$.*
   Note: Here $W^1[b+1, \ell_1]$ cannot be an empty walk because by case (C.2) b is a digon in W^1.

X. *otherwise, $\phi(\mathcal{W}) = W^1[1, a] \diamond \phi(W^1[a+1, \ell_1], W^2, \dots, W^p)$.*
   Note: The case C.X will form a "common case" with the case D.X.

D. if the vertex v^1_1 occurs in exactly two different walks:
Let i be the index of the another walk W^i in which v^1_1 occurs and let b be the index of the first occurrence of v^1_1 in W^i.

1. *if $r^1_1 = r^i_b = 0$, then $\phi(\mathcal{W}) = \mathcal{W} \leftrightarrow^{1,i}_{1,b}$.*
2. *if the index b is not a digon in W^i, then $\phi(\mathcal{W}) = \mathcal{W} \frown^{1,i}_{1,b}$.*
   Note: If neither case (1) nor (2) applies, then $r^1_1 \neq 0$.
3. *if v^1_1 occurs at least twice in W^i, then let c be the index of its second occurrence and $\phi(\mathcal{W}) = \mathcal{W} \leftrightarrow^{1,i}_{2,c+1}$.*
   Note: It can happen that one of the suffixes in this case is empty. However, both of them cannot be empty at the same time because W^1 and W^i have different ending vertices because $\mathcal{W}$ is proper.
   Note: In the remaining cases, v^1_1 occurs exactly once in W^i, and this occurrence is a digon at index b.
   Now, let a be the index of the last occurrence of v^1_1 in W^1 (if v^1_1 occurs only once in W^1, then $a = 1$).
4. *if $W^1[2, a-1]$ is not a palindrome, then $\phi(\mathcal{W}) = (W^1\overleftarrow{[2, a-1]}, W^2, \dots, W^p)$.*
   Note: If $a = 1$, then $W^1[2, a-1]$ is the empty walk which is a palindrome.
5. *if $a = \ell_1$, then $\phi(\mathcal{W}) = W^1 \sqcup \phi(W^2, \dots, W^p)$.*
6. *if $v^1_{a+1} = v^i_{b+1}$, then $\phi(\mathcal{W}) = \mathcal{W} \leftrightarrow^{1,i}_{a+1,b+1}$.*
   Note: By case (5) it holds that $a < \ell_1$ and by case (2) it holds that $b < \ell_i$.

X. *otherwise, $\phi(\mathcal{W}) = W^1[1, a] \diamond \phi(W^1[a+1, \ell_1], W^2, \dots, W^p)$.*
   Note: The case D.X will form a "common case" with the case C.X

Evaluating $f(\mathcal{C}_\ell)$

(3) $f(\mathcal{C}_\ell)$ can be evaluated in time $2^k n^{\mathcal{O}(1)}$

Evaluating $f(\mathcal{C}_\ell)$

(3) $f(\mathcal{C}_\ell)$ can be evaluated in time $2^k n^{\mathcal{O}(1)}$

- Dynamic programming that stores:

Evaluating $f(\mathcal{C}_\ell)$

(3) $f(\mathcal{C}_\ell)$ can be evaluated in time $2^k n^{\mathcal{O}(1)}$

- Dynamic programming that stores:
  - The length of the walk

Evaluating $f(\mathcal{C}_\ell)$

(3) $f(\mathcal{C}_\ell)$ can be evaluated in time $2^k n^{\mathcal{O}(1)}$

- Dynamic programming that stores:
  - The length of the walk
  - The last and the second last vertex of the walk

Evaluating $f(\mathcal{C}_\ell)$

(3) $f(\mathcal{C}_\ell)$ can be evaluated in time $2^k n^{\mathcal{O}(1)}$

- Dynamic programming that stores:
  - The length of the walk
  - The last and the second last vertex of the walk
  - Whether the last vertex is labeled

Evaluating $f(\mathcal{C}_\ell)$

(3) $f(\mathcal{C}_\ell)$ can be evaluated in time $2^k n^{\mathcal{O}(1)}$

- Dynamic programming that stores:
  - The length of the walk
  - The last and the second last vertex of the walk
  - Whether the last vertex is labeled
  - The set of used labels (2^k factor)

Evaluating $f(\mathcal{C}_\ell)$

(3) $f(\mathcal{C}_\ell)$ can be evaluated in time $2^k n^{\mathcal{O}(1)}$

- Dynamic programming that stores:
  - The length of the walk
  - The last and the second last vertex of the walk
  - Whether the last vertex is labeled
  - The set of used labels (2^k factor)

$\Rightarrow$ By DeMillo–Lipton–Schwartz–Zippel lemma, $2^k n^{\mathcal{O}(1)}$ time algorithm for k-colored (s, t)-path that works with high probability

Beyond maximum colored path

- Extending from single path to multiple paths requires new ideas

Beyond maximum colored path

- Extending from single path to multiple paths requires new ideas
  - New cancellation argument of swapping the suffixes of two intersecting walks

Beyond maximum colored path

- Extending from single path to multiple paths requires new ideas
  - New cancellation argument of swapping the suffixes of two intersecting walks
- Extending from colors to combination of weights and colors is easy

Beyond maximum colored path

- Extending from single path to multiple paths requires new ideas
  - New cancellation argument of swapping the suffixes of two intersecting walks
- Extending from colors to combination of weights and colors is easy
  - Instead of weights, could ask for any property that can be efficiently evaluated in DP

Conclusion

- We gave a $2^k n^{\mathcal{O}(1)}$ time algorithm for finding an (s, t)-path with at least k colors

Conclusion

- We gave a $2^k n^{\mathcal{O}(1)}$ time algorithm for finding an (s, t)-path with at least k colors
  - and $2^{k+p} n^{\mathcal{O}(1)}$ time algorithm for a more general setting with multiple paths and weights and colors

Conclusion

- We gave a $2^k n^{\mathcal{O}(1)}$ time algorithm for finding an (s, t)-path with at least k colors
  - and $2^{k+p} n^{\mathcal{O}(1)}$ time algorithm for a more general setting with multiple paths and weights and colors
- The proof that if no solution exists then $f(\mathcal{C}_\ell)$ is identically zero is the most complicated part

Conclusion

- We gave a $2^k n^{\mathcal{O}(1)}$ time algorithm for finding an (s,t)-path with at least k colors
  - and $2^{k+p} n^{\mathcal{O}(1)}$ time algorithm for a more general setting with multiple paths and weights and colors
- The proof that if no solution exists then $f(\mathcal{C}_\ell)$ is identically zero is the most complicated part
- Open problem: Is there an FPT-algorithm for maximum colored two disjoint paths?

Conclusion

- We gave a $2^k n^{\mathcal{O}(1)}$ time algorithm for finding an (s, t)-path with at least k colors
  - and $2^{k+p} n^{\mathcal{O}(1)}$ time algorithm for a more general setting with multiple paths and weights and colors
- The proof that if no solution exists then $f(\mathcal{C}_\ell)$ is identically zero is the most complicated part
- Open problem: Is there an FPT-algorithm for maximum colored two disjoint paths?
- Open problem: Is there a $1.99^k n^{\mathcal{O}(1)}$ time algorithm for longest (s, t)-path?

Thank you!

Thank you!