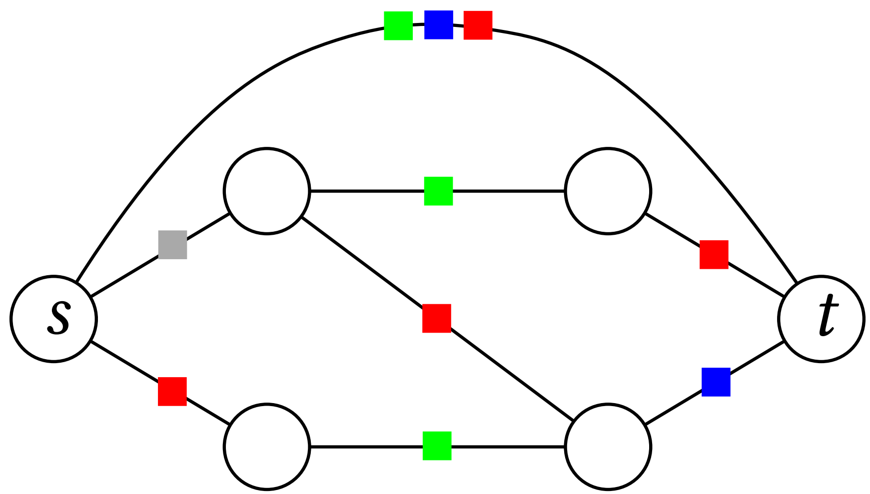
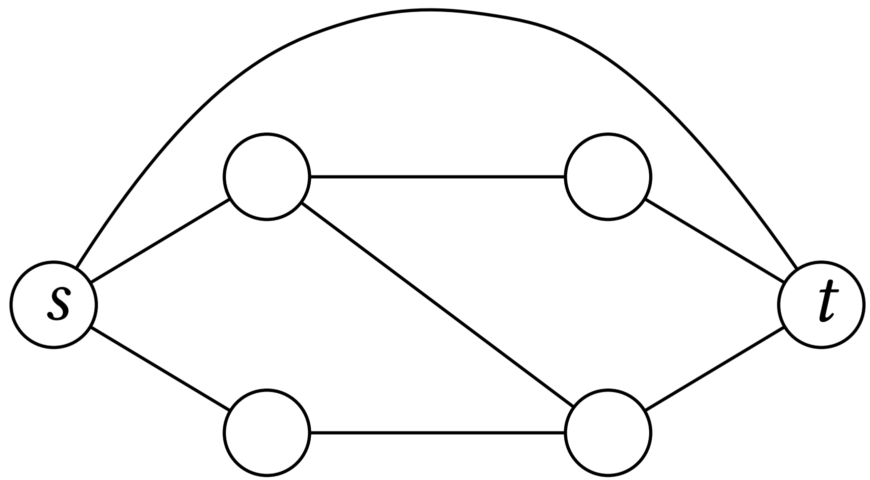


Computing a Minimum Color Path in Edge-Colored Graphs

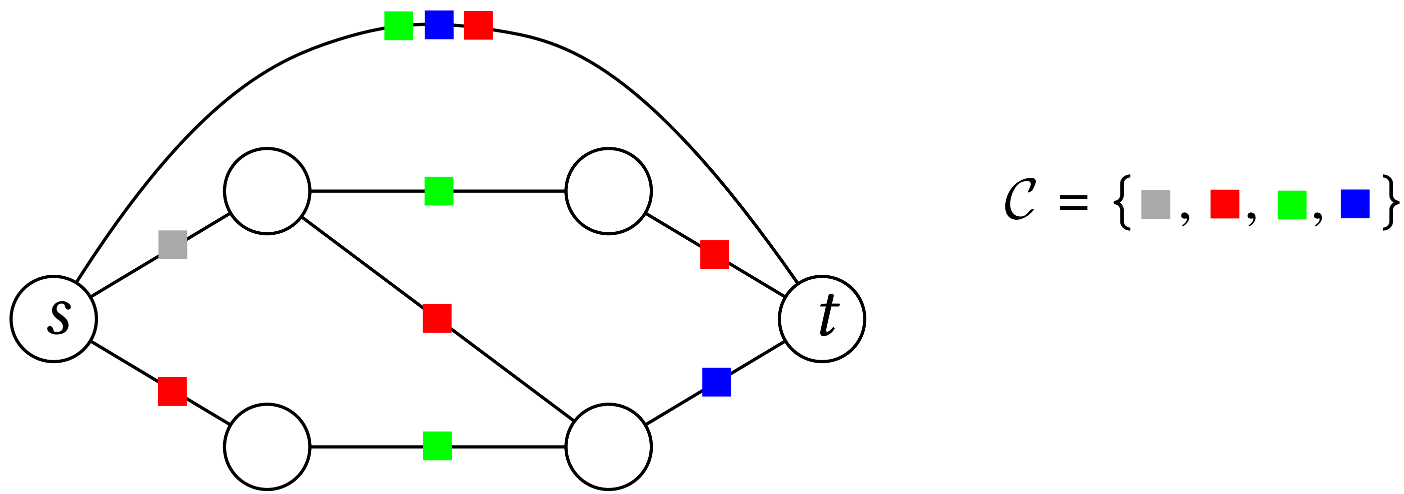
Neeraj Kumar
University of California, Santa Barbara

Problem Description



Input : A graph $G = (V, E)$, two fixed vertices s and t

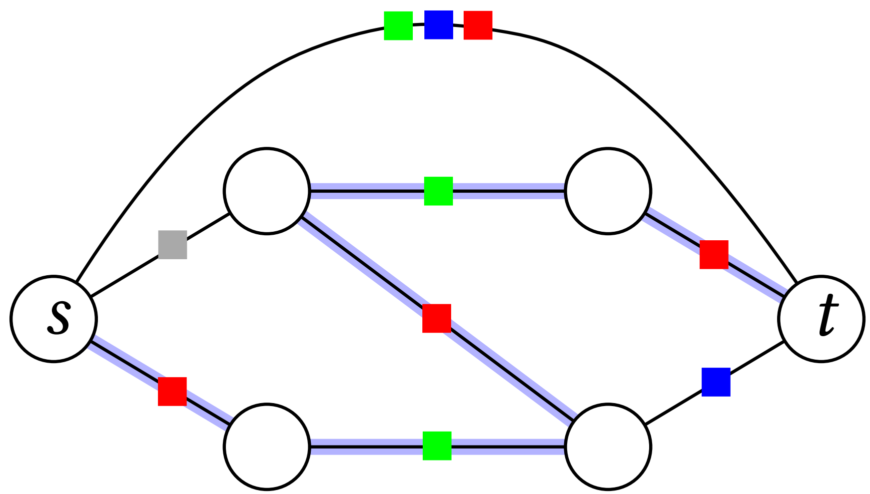
Problem Description



Input : A graph $G = (V, E)$, two fixed vertices s and t

A set of colors $\mathcal{C}$, and a mapping $\mathcal{X} : E \to 2^{\mathcal{C}}$

Problem Description



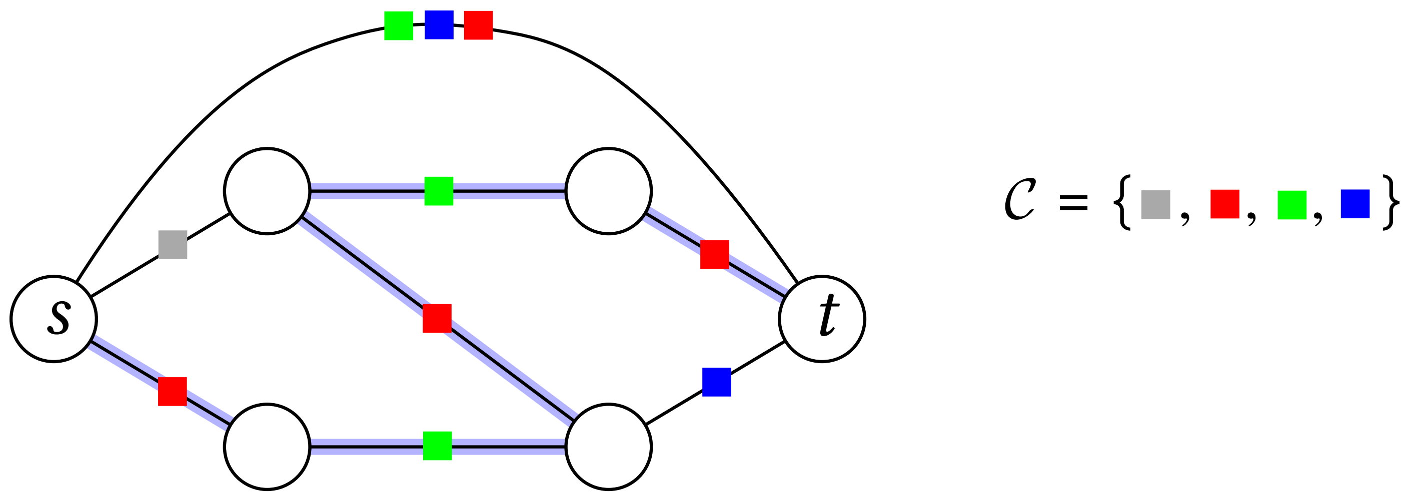
$\mathcal{C} = \{ \blacksquare, \blacksquare, \blacksquare, \blacksquare \}$

Input : A graph $G = (V, E)$, two fixed vertices s and t

A set of colors $\mathcal{C}$, and a mapping $\mathcal{X} : E \to 2^{\mathcal{C}}$

Find a path from s to t that "uses" fewest number of colors

Problem Description



Input : A graph $G = (V, E)$, two fixed vertices s and t

A set of colors $\mathcal{C}$, and a mapping $\mathcal{X} : E \to 2^{\mathcal{C}}$

Find a path from s to t that "uses" fewest number of colors

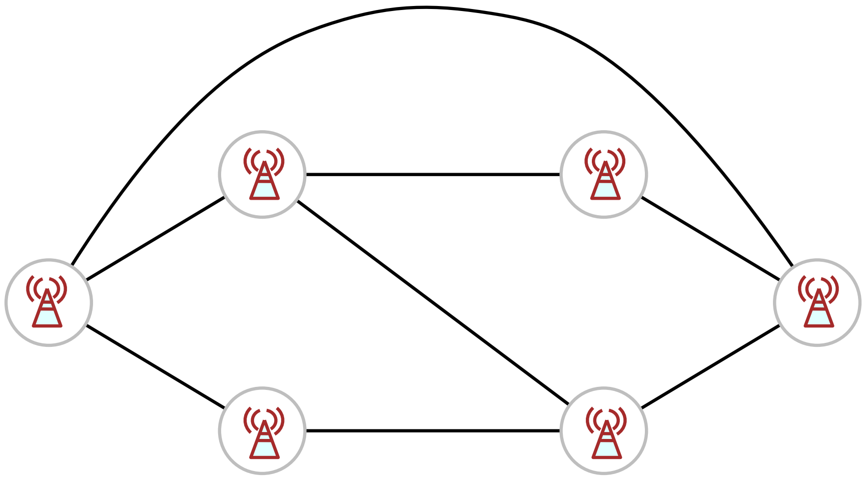
in other words, find a path π that minimizes $\left| \bigcup_{e \in \pi} \mathcal{X}(e) \right|$

Motivation

Reliability of connections in Mesh networks (Yuan et al., INFOCOM'05)

Motivation

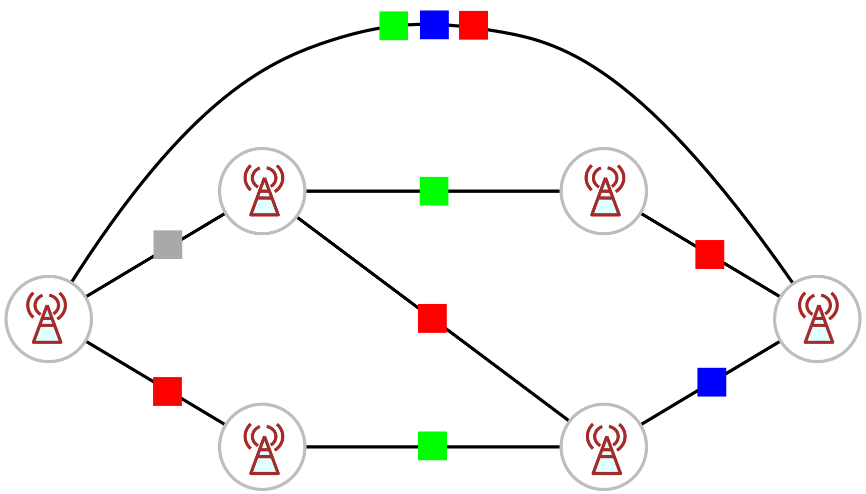
Reliability of connections in Mesh networks (Yuan et al., INFOCOM'05)

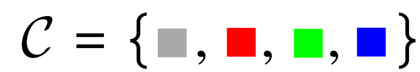


– Underlying network defines graph G

Motivation

Reliability of connections in Mesh networks (Yuan et al., INFOCOM'05)



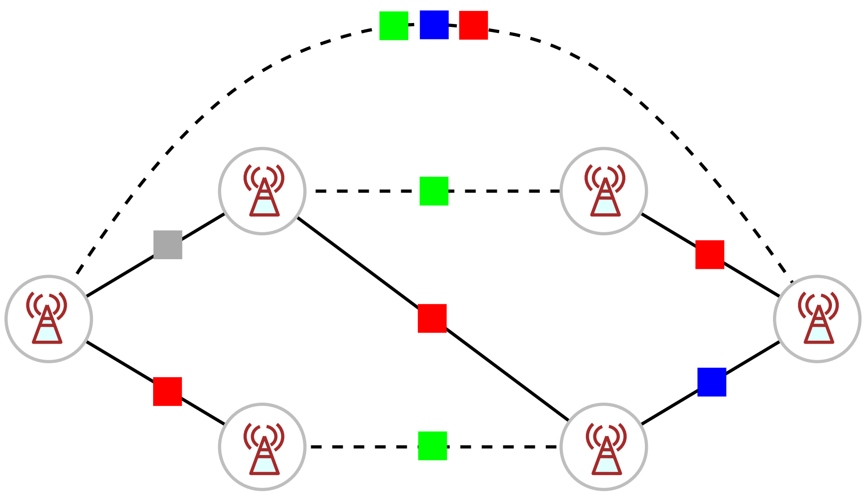


– Underlying network defines graph G

– Colors correspond to failure events

Motivation

Reliability of connections in Mesh networks (Yuan et al., INFOCOM'05)



$\mathcal{C} = \{$ ■, ■, ■, ■ $\}$

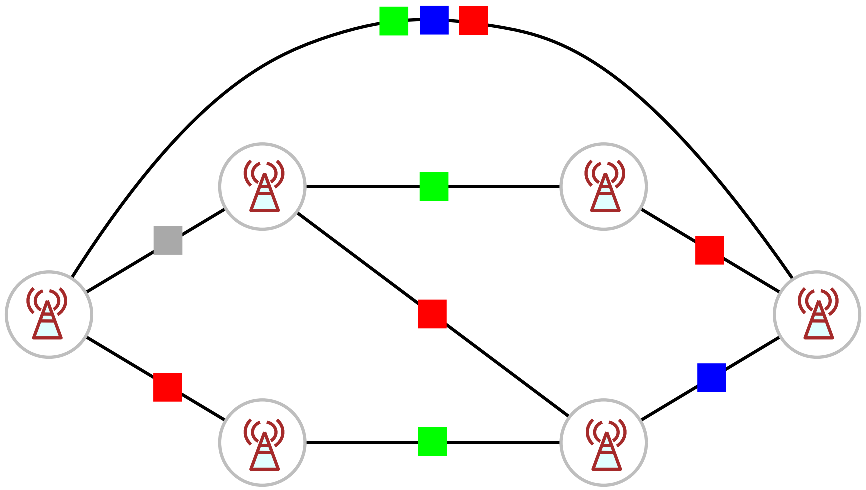
– Underlying network defines graph G

– Colors correspond to failure events

If event green ■ happens, all green links become unusable

Motivation

Reliability of connections in Mesh networks (Yuan et al., INFOCOM'05)



$\mathcal{C} = \{ \blacksquare, \blacksquare, \blacksquare, \blacksquare \}$

– Underlying network defines graph G

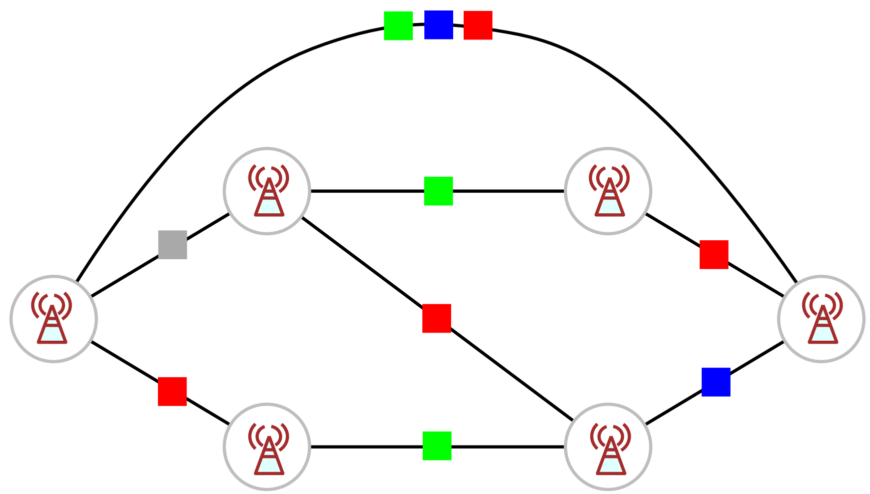
– Colors correspond to failure events

If event green ■ happens, all green links become unusable

If all events have same probabilities, a min-color path is most reliable!

Motivation

Reliability of connections in Mesh networks (Yuan et al., INFOCOM'05)



$\mathcal{C} = \{$ ■, ■, ■, ■ $\}$

– Underlying network defines graph G

– Colors correspond to failure events

If event green ■ happens, all green links become unusable

If all events have same probabilities, a min-color path is most reliable!

Other Applications : Color-Cost Model

colors are services, only pay for first use!

Network Licensing costs

Truck route planning

Previous Work

- Yuan et al.' 05 : NP-hardness and heuristic algorithms
  - Reduction from SET COVER : inapproximable within $\log n$

Previous Work

- Yuan et al.' 05 : NP-hardness and heuristic algorithms
  - Reduction from Set Cover : inapproximable within $\log n$

- O($\sqrt{n}$)-approximation on vertex- colored graphs

  [Bandyapadhyaya et al., APPROX'18]

  - Gives an $O(\sqrt{E})$-approximation for our problem

Previous Work

- Yuan et al.' 05 : NP-hardness and heuristic algorithms
  - Reduction from Set Cover : inapproximable within $\log n$

- $O(\sqrt{n})$-approximation on vertex- colored graphs
  [Bandyapadhyaya et al., APPROX'18]
  - Gives an $O(\sqrt{E})$-approximation for our problem

Interesting questions:

- Improved hardness guarantees? Conditional hardness of $o(n^{1/8})$

Previous Work

- Yuan et al.' 05 : NP-hardness and heuristic algorithms
  - Reduction from SET COVER : inapproximable within $\log n$
- $O(\sqrt{n})$-approximation on vertex- colored graphs [Bandyapadhyaya et al., APPROX'18]
  - Gives an $O(\sqrt{E})$-approximation for our problem

Interesting questions:

- Improved hardness guarantees? Conditional hardness of $o(n^{1/8})$
- Sublinear approximation? An $O(n^{2/3})$-approximation

Previous Work

- Yuan et al.' 05 : NP-hardness and heuristic algorithms
  - Reduction from Set Cover : inapproximable within $\log n$
- O($\sqrt{n}$)-approximation on vertex- colored graphs
  [Bandyapadhyaya et al., APPROX'18]
  - Gives an $O(\sqrt{E})$-approximation for our problem

Interesting questions:

- Improved hardness guarantees? — Conditional hardness of $o(n^{1/8})$
- Sublinear approximation? — An $O(n^{2/3})$-approximation
- Practical algorithms? — Two greedy strategies
  Significantly better performance than earlier algorithms

Summary of Results

- Hardness of approximation within a factor $O(n^{1/8})$

  Conditional hardness based on Dense Vs Random conjecture for densest k-subgraph problem

- An $O(n^{2/3})$-approximation Algorithm

  Builds upon approximation for vertex-colored graphs

- Two greedy heuristics: Greedy-Select and Greedy-Prune-Select

  Inspired from the above approximation algorithm

- Experimental evaluation and benchmark datasets

  Previous works used randomly generated colored graphs

Summary of Results

- Hardness of approximation within a factor $O(n^{1/8})$

  Conditional hardness based on Dense Vs Random conjecture for densest k-subgraph problem

- An $O(n^{2/3})$-approximation Algorithm

  Builds upon approximation for vertex-colored graphs

- Two greedy heuristics: Greedy-Select and Greedy-Prune-Select

  Inspired from the above approximation algorithm

- Experimental evaluation and benchmark datasets

  Previous works used randomly generated colored graphs

Rest of this talk

Backup : Hardness

Reduction from *Minimum k-union* problem

collection of m sets $S_1, \ldots, , S_m$, select k such that their union is minimized

Backup : Hardness

Reduction from *Minimum k-union* problem

collection of m sets $S_1, \ldots, , S_m$, select k such that their union is minimized

Conditionally hard to approximate within $O(n^{1/4})$ [Chlamatac et al., SODA'17]

Backup : Hardness

Reduction from *Minimum k-union* problem

collection of m sets $S_1, \ldots, , S_m$, select k such that their union is minimized

Conditionally hard to approximate within $O(n^{1/4})$ [Chlamatac et al., SODA'17]

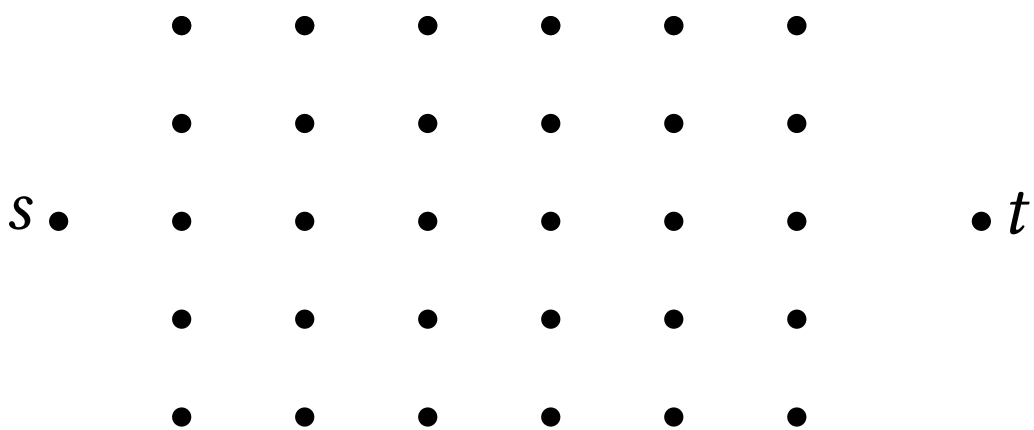
- Color set $\mathcal{C} = U$, the ground set of Minimum k-union

Backup : Hardness

Reduction from *Minimum k-union* problem

collection of m sets $S_1, \ldots, , S_m$, select k such that their union is minimized

Conditionally hard to approximate within $O(n^{1/4})$ [Chlamatac et al., SODA'17]



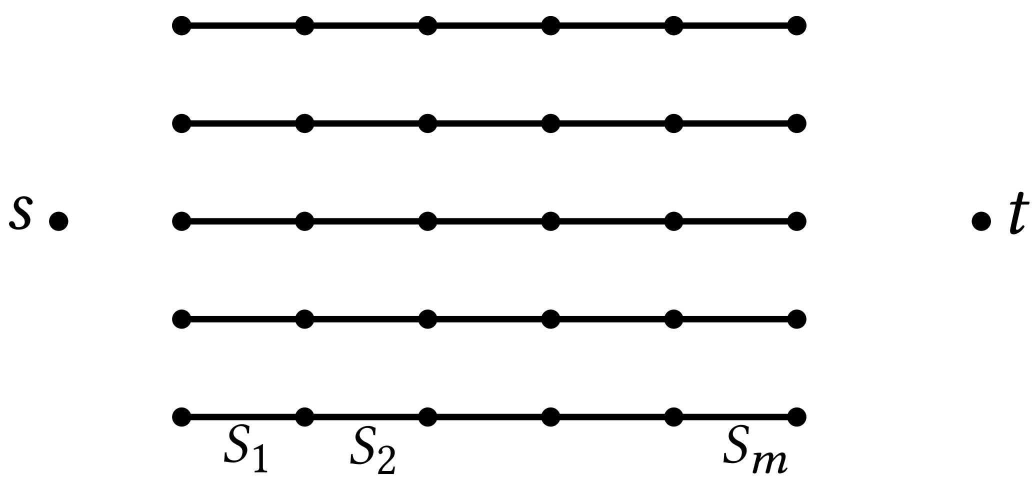
- Color set $\mathcal{C} = U$, the ground set of Minimum k-union
- Nodes arranged in a $(m + 1 - k) \times (m + 1)$-grid

Backup : Hardness

Reduction from *Minimum k-union* problem

collection of m sets $S_1, \ldots, , S_m$, select k such that their union is minimized

Conditionally hard to approximate within $O(n^{1/4})$ [Chlamatac et al., SODA'17]



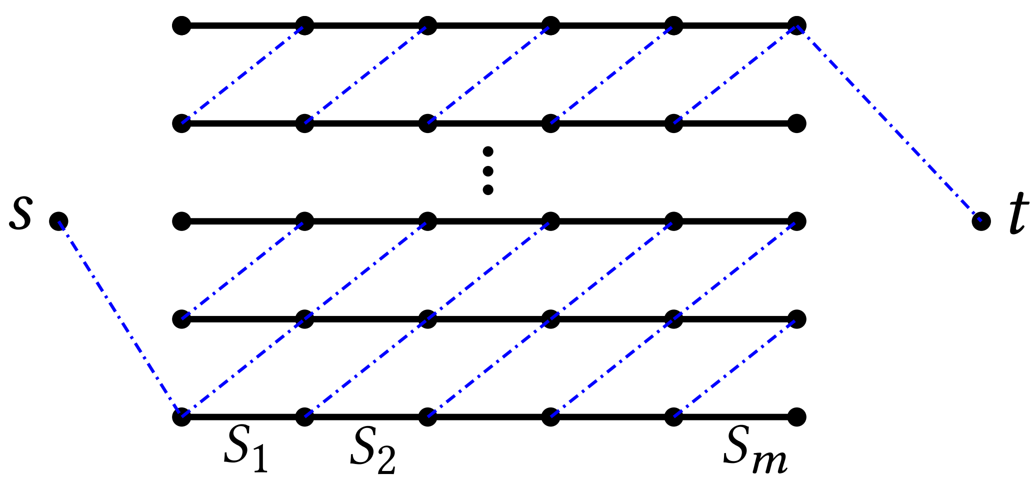
- Color set $\mathcal{C} = U$, the ground set of Minimum k-union
- Nodes arranged in a $(m + 1 - k) \times (m + 1)$-grid
- Horizontal edge between column $i, i + i$ is assigned colors S_i

Backup : Hardness

Reduction from *Minimum k-union* problem

collection of m sets $S_1, \ldots, , S_m$, select k such that their union is minimized

Conditionally hard to approximate within $O(n^{1/4})$ [Chlamatac et al., SODA'17]



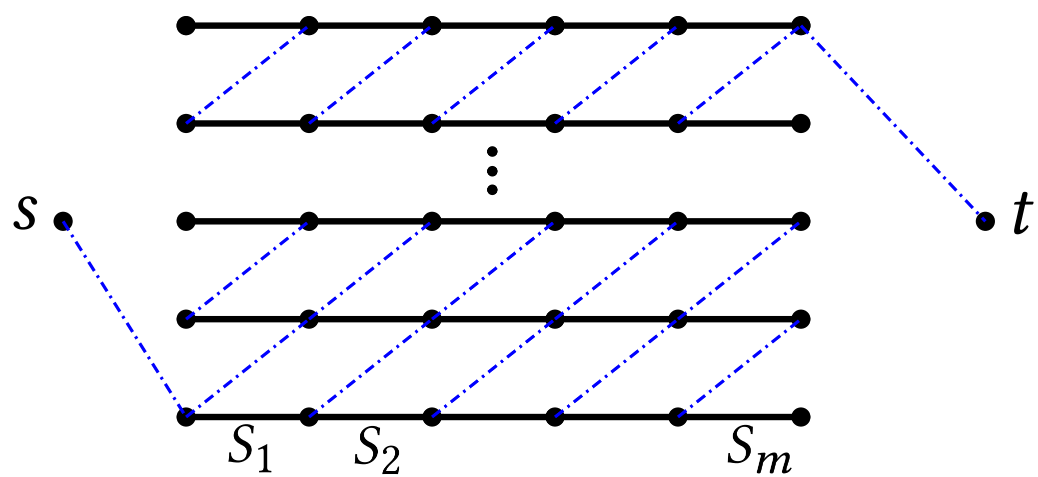
- Color set $\mathcal{C} = U$, the ground set of Minimum k-union
- Nodes arranged in a $(m + 1 - k) \times (m + 1)$-grid
- Horizontal edge between column $i, i + i$ is assigned colors S_i
- Add diagonal edges that are free (empty colorset)

Backup : Hardness

Reduction from *Minimum k-union* problem

collection of m sets $S_1, \ldots, , S_m$, select k such that their union is minimized

Conditionally hard to approximate within $O(n^{1/4})$ [Chlamatac et al., SODA'17]



horizontal displacement m

vertical displacement $m - k$

- Color set $\mathcal{C} = U$, the ground set of Minimum k-union
- Nodes arranged in a $(m + 1 - k) \times (m + 1)$-grid
- Horizontal edge between column $i, i + i$ is assigned colors S_i
- Add diagonal edges that are free (empty colorset)

Minimum k-union corresponds to a min-color path in G

Approximation Algorithm

Input: A colored graph $G = (V, E, \mathcal{C}, \mathcal{X})$,

Approximation Algorithm

Input: A colored graph $G = (V, E, \mathcal{C}, \mathcal{X})$, and an integer k

compute a path that uses at most k-colors

Approximation Algorithm

Input: A colored graph $G = (V, E, \mathcal{C}, \mathcal{X})$, and an integer k
compute a path that uses at most k-colors

- Remove all edges that have more than k colors

Approximation Algorithm

Input: A colored graph $G = (V, E, \mathcal{C}, \mathcal{X})$, and an integer k
compute a path that uses at most k-colors

- Remove all edges that have more than k colors
- A path of length ℓ uses at most $k\ell$ colors

Approximation Algorithm

Input: A colored graph $G = (V, E, \mathcal{C}, \mathcal{X})$, and an integer k
compute a path that uses at most k-colors

- Remove all edges that have more than k colors
- A path of length ℓ uses at most $k\ell$ colors
  - If G has "small" diameter (dense), any path is "good" approximation
  - Otherwise, G has fewer edges, can use ideas from $O(\sqrt{E})$-approximation

Approximation Algorithm

Input: A colored graph $G = (V, E, \mathcal{C}, \mathcal{X})$, and an integer k
compute a path that uses at most k-colors

- Remove all edges that have more than k colors
- A path of length ℓ uses at most $k\ell$ colors
  - If G has "small" diameter (dense), any path is "good" approximation
  - Otherwise, G has fewer edges, can use ideas from $O(\sqrt{E})$-approximation

Partitioning vertices of G as dense and sparse

Approximation Algorithm

Input: A colored graph $G = (V, E, \mathcal{C}, \mathcal{X})$, and an integer k
compute a path that uses at most k-colors

- Remove all edges that have more than k colors
- A path of length ℓ uses at most $k\ell$ colors
  - If G has “small” diameter (dense), any path is "good" approximation
  - Otherwise, G has fewer edges, can use ideas from $O(\sqrt{E})$-approximation

Partitioning vertices of G as dense and sparse

– Recursively pick a vertex of degree at most β, mark it as *sparse* and remove it

– Mark remaining vertices as *dense*

Approximation Algorithm

Input: A colored graph $G = (V, E, \mathcal{C}, \mathcal{X})$, and an integer k

compute a path that uses at most k-colors

- Remove all edges that have more than k colors
- A path of length ℓ uses at most $k\ell$ colors
  - If G has "small" diameter (dense), any path is "good" approximation
  - Otherwise, G has fewer edges, can use ideas from $O(\sqrt{E})$-approximation

Partitioning vertices of G as dense and sparse

– Recursively pick a vertex of degree at most β, mark it as *sparse* and remove it

– Mark remaining vertices as *dense*

diameter of a connected dense component $C_i = \frac{\text{size of } C_i}{\text{min-degree}} \leq \frac{n_i}{\beta}$

Approximation Steps

An edge $e = (u, v)$ is dense if both u, v are dense, otherwise e is sparse

Approximation Steps

An edge $e = (u, v)$ is dense if both u, v are dense, otherwise e is sparse

- *Un-color* every dense edge $e = (u, v)$, that is, set $\mathcal{X}(e) = \varnothing$

Approximation Steps

An edge $e = (u, v)$ is dense if both u, v are dense, otherwise e is sparse

- *Un-color* every dense edge $e = (u, v)$, that is, set $\mathcal{X}(e) = \varnothing$
- *At this point, there are at most $kn\beta$ occurrences of all colors

  Why : At most $n\beta$ sparse edges with at most k colors each

Approximation Steps

An edge $e = (u, v)$ is dense if both u, v are dense, otherwise e is sparse

- *Un-color* every dense edge $e = (u, v)$, that is, set $\mathcal{X}(e) = \varnothing$
- *A*t this point, there are at most $kn\beta$ occurrences of all colors

  Why : At most $n\beta$ sparse edges with at most k colors each
- *R*emove colors that occur on at most $\sqrt{n\beta}$ edges

  At most $k\sqrt{n\beta}$ such colors

Approximation Steps

An edge $e = (u, v)$ is dense if both u, v are dense, otherwise e is sparse

- *Un-color* every dense edge $e = (u, v)$, that is, set $\mathcal{X}(e) = \varnothing$
- *At* this point, there are at most $kn\beta$ occurrences of all colors

  Why : At most $n\beta$ sparse edges with at most k colors each
- *R*emove colors that occur on at most $\sqrt{n\beta}$ edges

  At most $k\sqrt{n\beta}$ such colors
- *R*un shortest path algorithm with cardinality $|\mathcal{X}(e)|$ as weight

  Computes a path π that uses at most $k\sqrt{n\beta}$ colors

Approximation Steps

An edge $e = (u, v)$ is dense if both u, v are dense, otherwise e is sparse

- *Un-color* every dense edge $e = (u, v)$, that is, set $\mathcal{X}(e) = \emptyset$
- *A*t this point, there are at most $kn\beta$ occurrences of all colors

  Why : At most $n\beta$ sparse edges with at most k colors each

- *R*emove colors that occur on at most $\sqrt{n\beta}$ edges

  At most $k\sqrt{n\beta}$ such colors

- *R*un shortest path algorithm with cardinality $|\mathcal{X}(e)|$ as weight

  Computes a path π that uses at most $k\sqrt{n\beta}$ colors

Colors used by π in original graph $\leq \sum_i \frac{kn_i}{\beta} + 2k\sqrt{n\beta}$

Approximation Steps

An edge $e = (u, v)$ is dense if both u, v are dense, otherwise e is sparse

- *Un-color* every dense edge $e = (u, v)$, that is, set $\mathcal{X}(e) = \varnothing$
- *A*t this point, there are at most $kn\beta$ occurrences of all colors

  Why : At most $n\beta$ sparse edges with at most k colors each
- *R*emove colors that occur on at most $\sqrt{n\beta}$ edges

  At most $k\sqrt{n\beta}$ such colors
- *R*un shortest path algorithm with cardinality $|\mathcal{X}(e)|$ as weight

  Computes a path π that uses at most $k\sqrt{n\beta}$ colors

Colors used by π in original graph $\leq \boxed{\sum_i \frac{kn_i}{\beta}} + 2k\sqrt{n\beta}$

π enters each connected dense component at most once

Approximation Steps

An edge $e = (u, v)$ is dense if both u, v are dense, otherwise e is sparse

- *Un-color* every dense edge $e = (u, v)$, that is, set $\mathcal{X}(e) = \emptyset$
- *A*t this point, there are at most $kn\beta$ occurrences of all colors

  Why : At most $n\beta$ sparse edges with at most k colors each
- *R*emove colors that occur on at most $\sqrt{n\beta}$ edges

  At most $k\sqrt{n\beta}$ such colors
- *R*un shortest path algorithm with cardinality $|\mathcal{X}(e)|$ as weight

  Computes a path π that uses at most $k\sqrt{n\beta}$ colors

Colors used by π in original graph $\leq \boxed{\sum_i \frac{kn_i}{\beta}} + 2k\sqrt{n\beta}$

π enters each connected dense component at most once

For $\beta = n^{1/3}$, we obtain a path π that uses at most $O(n^{2/3}) \cdot k$ colors

Greedy Strategies

Strategy 1: Guess a color that the optimal path is likely to use

Greedy Strategies

Strategy 1: Guess a color that the optimal path is likely to use

A reasonably long path must repeat a lot of its colors.

If a color appears too many times in G, it is likely to repeat on π as well

Greedy Strategies

Strategy 1: Guess a color that the optimal path is likely to use

A reasonably long path must repeat a lot of its colors.

If a color appears too many times in G, it is likely to repeat on π as well

GREEDY-SELECT

- On iteration i , select color $c_{\max}$ that occurs on max number of edges of G_{i-1}

Greedy Strategies

Strategy 1: Guess a color that the optimal path is likely to use

A reasonably long path must repeat a lot of its colors.

If a color appears too many times in G, it is likely to repeat on π as well

Greedy-Select

- On iteration i , select color $c_{\max}$ that occurs on max number of edges of G_{i-1}
- Remove all occurrences of $c_{\max}$ from G_{i-1} to obtain G_i

Greedy Strategies

Strategy 1: Guess a color that the optimal path is likely to use

A reasonably long path must repeat a lot of its colors.

If a color appears too many times in G, it is likely to repeat on π as well

GREEDY-SELECT

- On iteration i , select color $c_{\max}$ that occurs on max number of edges of G_{i-1}
- Remove all occurrences of $c_{\max}$ from G_{i-1} to obtain G_i
- Compute the shortest path π_i in G_i with $w(e) = |\mathcal{X}(e)|$

Greedy Strategies

Strategy 1: Guess a color that the optimal path is likely to use

A reasonably long path must repeat a lot of its colors.

If a color appears too many times in G, it is likely to repeat on π as well

Greedy-Select

- On iteration i , select color $c_{\max}$ that occurs on max number of edges of G_{i-1}
- Remove all occurrences of $c_{\max}$ from G_{i-1} to obtain G_i
- Compute the shortest path π_i in G_i with $w(e) = |\mathcal{X}(e)|$

Return the best path π_i, in terms of number of colors of orginal graph $G_0 = G$

Greedy Strategies

Strategy 1: Guess a color that the optimal path is likely to use

A reasonably long path must repeat a lot of its colors.

If a color appears too many times in G, it is likely to repeat on π as well

GREEDY-SELECT

- On iteration i , select color $c_{\max}$ that occurs on max number of edges of G_{i-1}
- Remove all occurrences of $c_{\max}$ from G_{i-1} to obtain G_i
- Compute the shortest path π_i in G_i with $w(e) = |\mathcal{X}(e)|$

Return the best path π_i, in terms of number of colors of orginal graph $G_0 = G$

Strategy 2: Guess a color that the optimal path is *not likely* to use

GREEDY-PRUNE

Greedy Strategies

Strategy 1: Guess a color that the optimal path is likely to use

A reasonably long path must repeat a lot of its colors.

If a color appears too many times in G, it is likely to repeat on π as well

Greedy-Select

- On iteration i , select color $c_{\max}$ that occurs on max number of edges of G_{i-1}
- Remove all occurrences of $c_{\max}$ from G_{i-1} to obtain G_i
- Compute the shortest path π_i in G_i with $w(e) = |\mathcal{X}(e)|$

Return the best path π_i, in terms of number of colors of orginal graph $G_0 = G$

Strategy 2: Guess a color that the optimal path is *not likely* to use

Greedy-Prune

–Select a color $c_{\min}$ that occurs on fewest number of edges of G

– Remove all edges containing $c_{\min}$ provided s, t are still connected

(Also edges that do not lie in same component as s, t)

Greedy Strategies

Strategy 1: Guess a color that the optimal path is likely to use

A reasonably long path must repeat a lot of its colors.

If a color appears too many times in G, it is likely to repeat on π as well

GREEDY-SELECT

- On iteration i , select color $c_{\max}$ that occurs on max number of edges of G_{i-1}
- Remove all occurrences of $c_{\max}$ from G_{i-1} to obtain G_i
- Compute the shortest path π_i in G_i with $w(e) = |\mathcal{X}(e)|$

Return the best path π_i, in terms of number of colors of orginal graph $G_0 = G$

Strategy 2: Guess a color that the optimal path is *not likely* to use

GREEDY-PRUNE

- Select a color $c_{\min}$ that occurs on fewest number of edges of G
- Remove all edges containing $c_{\min}$ provided s, t are still connected

(Also edges that do not lie in same component as s, t)

Let G' be the graph obtained from above. Return shortest path in G'

Greedy Strategies

Observe that GREEDY-PRUNE makes the graph sparser

Greedy Strategies

Observe that Greedy-Prune makes the graph sparser

- Can run Greedy-Select over this sparse graph

Helps the path by *not choosing* colors that occur small number of times
(unless absolutely necessary)

Greedy Strategies

Observe that GREEDY-PRUNE makes the graph sparser

– Can run GREEDY-SELECT over this sparse graph

Helps the path by *not choosing* colors that occur small number of times

(unless absolutely necessary)

GREEDY-PRUNE-SELECT

– Set *threshold* to say $|E|/c$

Greedy Strategies

Observe that Greedy-Prune makes the graph sparser

– Can run Greedy-Select over this sparse graph

Helps the path by *not choosing* colors that occur small number of times (unless absolutely necessary)

Greedy-Prune-Select

– Set *threshold* to say $|E|/c$

– Run Greedy-Prune to make the graph sparser

Greedy Strategies

Observe that GREEDY-PRUNE makes the graph sparser

– Can run GREEDY-SELECT over this sparse graph

Helps the path by *not choosing* colors that occur small number of times
(unless absolutely necessary)

GREEDY-PRUNE-SELECT

– Set *threshold* to say $|E|/c$

– Run GREEDY-PRUNE to make the graph sparser

– If G changes significantly (more than *threshold* edges deleted) :
run GREEDY-SELECT

(at most c calls of GREEDY-SELECT)

Greedy Strategies

Observe that GREEDY-PRUNE makes the graph sparser
– Can run GREEDY-SELECT over this sparse graph

Helps the path by *not choosing* colors that occur small number of times
(unless absolutely necessary)

GREEDY-PRUNE-SELECT

– Set *threshold* to say $|E|/c$

– Run GREEDY-PRUNE to make the graph sparser

– If G changes significantly (more than *threshold* edges deleted) :
run GREEDY-SELECT (at most c calls of GREEDY-SELECT)

– Return the best path found

$c = 4$ seems to gives a good compromise between path quality and runtime

Datasets

Previous works have used *random colored graphs*

$G = (n, p)$ and each edge is indep. assigned one color uniformly from the set $\mathcal{C}$

Datasets

Previous works have used *random colored graphs*

$G = (n, p)$ and each edge is indep. assigned one color uniformly from the set $\mathcal{C}$

Two major concerns:

– application scenario connectivity is typically not random

– have small diameters, small difference between dijkstra and optimal w.h.p

(less room for improvement)

Datasets

Previous works have used *random colored graphs*

$G = (n, p)$ and each edge is indep. assigned one color uniformly from the set $\mathcal{C}$

Two major concerns:

– application scenario connectivity is typically not random

– have small diameters, small difference between dijkstra and optimal w.h.p

(less room for improvement)

Instance	Dijkstra	Best known	Remarks
Layered	43.38	17.6	$k = 4$ nodes per layer
Unit-disk	34.66	13.88	$n = 1000$ random disks in a 10×100 rectangle
Road-network	366	246	$1.5M$ nodes, $2.7M$ edges, 500 colors
Uniform-col [19]	11.45	9.5	edges added with $p = \log n / 2n$

Datasets

Previous works have used *random colored graphs*

$G = (n, p)$ and each edge is indep. assigned one color uniformly from the set $\mathcal{C}$

Two major concerns:

– application scenario connectivity is typically not random

– have small diameters, small difference between dijkstra and optimal w.h.p

(less room for improvement)

Instance	Dijkstra	Best known	Remarks
Layered	43.38	17.6	$k = 4$ nodes per layer
Unit-disk	34.66	13.88	$n = 1000$ random disks in a 10×100 rectangle
Road-network	366	246	$1.5M$ nodes, $2.7M$ edges, 500 colors
Uniform-col [19]	11.45	9.5	edges added with $p = \log n/2n$

Need to construct more challenging instances!

Constructing Challenging Instances

An instance is challenging if

- expected number of colors on any $s - t$ path is large
- there exists a path with small number of colors

Constructing Challenging Instances

An instance is challenging if

- expected number of colors on any $s - t$ path is large
- there exists a path with small number of colors

If we assign colors indep. with uniform probability $\frac{1}{|\mathcal{C}|}$

For any path π of length ℓ, $p_{i\pi} = 1 - \left(1 - \frac{1}{|\mathcal{C}|}\right)^{\ell}$ (prob. that π uses color i)

Expected number of colors = $|\mathcal{C}| p_{i\pi} \propto \ell$

Constructing Challenging Instances

An instance is challenging if

- expected number of colors on any $s - t$ path is large
- there exists a path with small number of colors

If we assign colors indep. with uniform probability $\frac{1}{|\mathcal{C}|}$

For any path π of length ℓ, $p_{i\pi} = 1 - \left(1 - \frac{1}{|\mathcal{C}|}\right)^{\ell}$ (prob. that π uses color i)

Expected number of colors = $|\mathcal{C}| p_{i\pi} \propto \ell$

Probability that π contains k colors = $\binom{|\mathcal{C}|}{k} \cdot p_{i\pi}^{k} \cdot (1 - p_{i\pi})^{|\mathcal{C}|-k}$

Prob of k successes in binomial trial $B(|\mathcal{C}|, p_{i\pi})$ (typically a small value if k is small w.r.t $|\mathcal{C}|$)

Constructing Challenging Instances

An instance is challenging if

- expected number of colors on any $s - t$ path is large (*s*, *t* far apart)
- there exists a path with small number of colors (locally dense)

If we assign colors indep. with uniform probability $\frac{1}{|\mathcal{C}|}$

For any path π of length ℓ, $p_{i\pi} = 1 - \left(1 - \frac{1}{|\mathcal{C}|}\right)^{\ell}$ (prob. that π uses color i)

$$\text{Expected number of colors} = |\mathcal{C}| p_{i\pi} \propto \ell$$

$$\text{Probability that } \pi \text{ contains } k \text{ colors} = \binom{|\mathcal{C}|}{k} \cdot p_{i\pi}^{k} \cdot (1 - p_{i\pi})^{|\mathcal{C}|-k}$$

Prob of k successes in binomial trial $B(|\mathcal{C}|, p_{i\pi})$ (typically a small value if k is small w.r.t $|\mathcal{C}|$)

Takeway: $\boxed{s \text{ and } t \text{ should be far apart}}$ with a $\boxed{\text{large number of } s - t \text{ paths}}$

Constructing Challenging Instances

An instance is challenging if

- expected number of colors on any $s - t$ path is large ← (s, t far apart)
- there exists a path with small number of colors ← (locally dense)

If we assign colors indep. with uniform probability $\frac{1}{|\mathcal{C}|}$

For any path π of length ℓ, $\quad p_{i\pi} = 1 - \left(1 - \frac{1}{|\mathcal{C}|}\right)^{\ell} \quad$ (prob. that π uses color i)

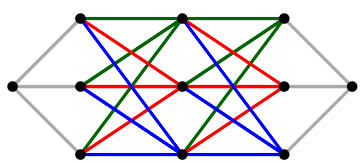
Expected number of colors = $|\mathcal{C}| p_{i\pi} \propto \ell$

Probability that π contains k colors = $\binom{|\mathcal{C}|}{k} \cdot p_{i\pi}^{k} \cdot (1 - p_{i\pi})^{|\mathcal{C}|-k}$

Prob of k successes in binomial trial $B(|\mathcal{C}|, p_{i\pi})$ (typically a small value if k is small w.r.t $|\mathcal{C}|$)

Takeway: $\boxed{s \text{ and } t \text{ should be far apart}}$ with a $\boxed{\text{large number of } s - t \text{ paths}}$

- Unit disk intersection graphs
- Layered Graphs
- CA Road network datatset
- Internet topology datatset (of ISP providers, `topology-zoo.org`)



Experimental Results

Number of colors = 50 (resp. 500) on small (resp. large) instances,

Table 2. Path quality and running time on layered graph instances.

Algorithm	Colors used	Time taken (ms)	Colors used	Time taken (ms)
	$4 \times 125 = 0.5k$ nodes		$4 \times 2500 = 10k$ nodes	
Dijkstra	36.8	0.6	441.8	23.6
SPACOA	33.6	65	396	127×10^3
Greedy-Select	18.2	12.6	185.6	3.5×10^3
Greedy-Prune-Select	17.2	49	173	12.5×10^3
ILP	16.4	707×10^3	∞	∞

Table : Layered graph instances

SPACOA is best previous heuristic

Experimental Results

Number of colors = 50 (resp. 500) on small (resp. large) instances,

Table 2. Path quality and running time on layered graph instances.

Algorithm	Colors used	Time taken (ms)	Colors used	Time taken (ms)
	$4 \times 125 = 0.5k$ nodes		$4 \times 2500 = 10k$ nodes	
Dijkstra	36.8	0.6	441.8	23.6
SPACOA	33.6	65	396	127×10^3
Greedy-Select	18.2	12.6	185.6	3.5×10^3
Greedy-Prune-Select	17.2	49	173	12.5×10^3
ILP	16.4	707×10^3	∞	∞

Table : Layered graph instances

Table 3. Path quality and running time on Unit disk graph instances.

Algorithm	Colors used	Time taken (ms)	Colors used	Time taken (ms)
	$4 \times 125 = 0.5k$ nodes		$4 \times 2500 = 10k$ nodes	
Dijkstra	28.8	1	357.8	38
SPACOA	23	124.8	333.6	41.4×10^3
Greedy-Select	14.2	13	145.6	4.7×10^3
Greedy-Prune-Select	13.4	55	134	17.6×10^3
ILP	12.6	1176×10^3	∞	∞

Table : Unit-disk Graph instances

SPACOA is best previous heuristic

Results Contd.

Table 4. Path quality and running time on some real world instances.

Algorithm	Colors used	Time taken (ms)	Colors used	Time taken (ms)
	CA Road Network		Internet topology	
Dijkstra	366	3.068×10^3	7	26
SPACOA	355	3.12×10^6	4	3111
Greedy-Select	251	0.73×10^6	5	29
Greedy-Prune-Select	246	2.71×10^6	4	286
ILP	∞	∞	4	1817

CA Road network 1.5 M nodes, 2.7M edges, 500 colors randomly assigned to edges

Internet topology 5.6k nodes, 8.6k edges, 261 colors (each color is a service provider)

Table 5. Path quality and running time on Uniform-Col instances.

Algorithm	Colors used	Time taken (in ms)	Colors used	Time taken (in ms)
	10^3 nodes		10^4 nodes	
Dijkstra	11.45	0.95	11.7	15
SPACOA	9.95	75.45	11.2	8664
Greedy-Select	10.4	9.2	11.5	150
Greedy-Prune-Select	10.3	46.3	11.4	1919
ILP	9.5	3913.8	-	-

Table: Random Colored Graphs

Summary of Results

- Hardness of approximation within a factor $O(n^{1/8})$
- An $O(n^{2/3})$-approximation Algorithm
- Two greedy heuristics: Greedy-Select and Greedy-Prune-Select
- Experimental evaluation and benchmark datasets

Thanks!

Summary of Results

- Hardness of approximation within a factor $O(n^{1/8})$
- An $O(n^{2/3})$-approximation Algorithm
- Two greedy heuristics: Greedy-Select and Greedy-Prune-Select
- Experimental evaluation and benchmark datasets

Thanks!

Rest of this talk