Longest Path Problem

Ahnaf Faisal, 1505005
Raihanul Alam, 1505010
Mahim Mahbub, 1505022
Zahin Wahab, 1505031
Bishal Basak Papan, 1505043

Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology

November 9, 2020

Where we left off

- We have already talked about the **Longest Path Problem** in our previous presentations.

Where we left off

- We have already talked about the **Longest Path Problem** in our previous presentations.
- We analysed the hardness of this problem and showed that to solve the problem exactly, we need **exponential** time.

Where we left off

- We have already talked about the **Longest Path Problem** in our previous presentations.
- We analysed the hardness of this problem and showed that to solve the problem exactly, we need **exponential** time.
- But before we proceed any further, let's just shed some light to our previous discussion.

What is a Longest Path Problem?

Optimization Version

Given a weighted graph G, find a simple path in this graph which has the maximum weight.

But Longest Path problem is NP-Complete

- From our previous discussion, we can safely state that Longest Path problem is **NP-Complete** which makes it both **NP** and **NP-Hard**.
- So, unless **P=NP**, there is no polynomial time algorithm which gives exact solution of Longest Path problem.
- But solving a **NP-hard optimization problem** like ours optimally takes a toll on running time. So we are going to relax the criterion of getting an optimal solution.
- Thus we will try to find an algorithm which solves the aforementioned problem approximately in polynomial time. This brings us to this week's content : **Approximation Algorithms**.

But Longest Path problem is NP-Complete

- From our previous discussion, we can safely state that Longest Path problem is **NP-Complete** which makes it both **NP** and **NP-Hard**.
- So, unless **P=NP**, there is no polynomial time algorithm which gives exact solution of Longest Path problem.
- But solving a **NP-hard optimization problem** like ours optimally takes a toll on running time. So we are going to relax the criterion of getting an optimal solution.
- Thus we will try to find an algorithm which solves the aforementioned problem approximately in polynomial time. This brings us to this week's content : **Approximation Algorithms**.

But Longest Path problem is NP-Complete

- From our previous discussion, we can safely state that Longest Path problem is **NP-Complete** which makes it both **NP** and **NP-Hard**.
- So, unless **P=NP**, there is no polynomial time algorithm which gives exact solution of Longest Path problem.
- But solving a **NP-hard optimization problem** like ours optimally takes a toll on running time. So we are going to relax the criterion of getting an optimal solution.
- Thus we will try to find an algorithm which solves the aforementioned problem approximately in polynomial time. This brings us to this week's content : **Approximation Algorithms**.

But Longest Path problem is NP-Complete

- From our previous discussion, we can safely state that Longest Path problem is **NP-Complete** which makes it both **NP** and **NP-Hard**.
- So, unless **P=NP**, there is no polynomial time algorithm which gives exact solution of Longest Path problem.
- But solving a **NP-hard optimization problem** like ours optimally takes a toll on running time. So we are going to relax the criterion of getting an optimal solution.
- Thus we will try to find an algorithm which solves the aforementioned problem approximately in polynomial time. This brings us to this week's content : **Approximation Algorithms**.

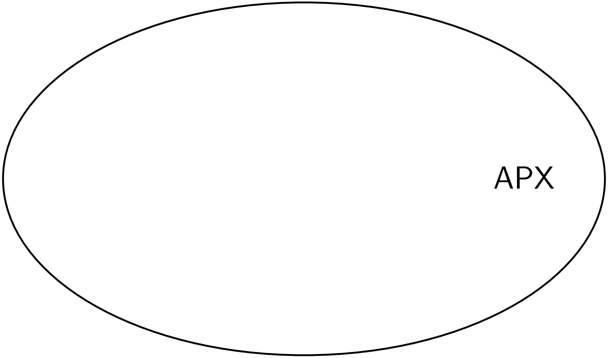
Our take on Approximation Algorithms

- We will show that our problem does not have a constant factor approximation algorithm, unless P=NP. That is our problem does not lie in **APX-class**.

Approximation classes

CLASS APX

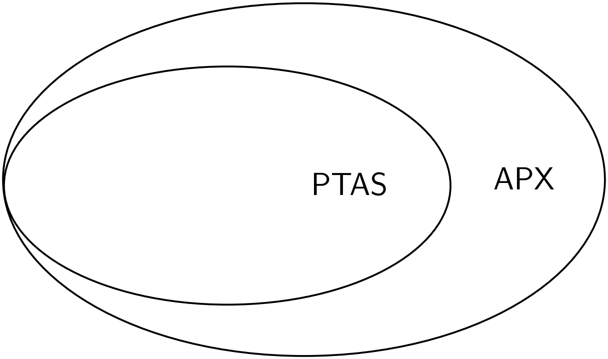
The class **APX (an abbreviation of "approximable")** is the set of **NP optimization problems** that allow polynomial-time constant-factor approximation algorithms.



Approximation classes

CLASS PTAS

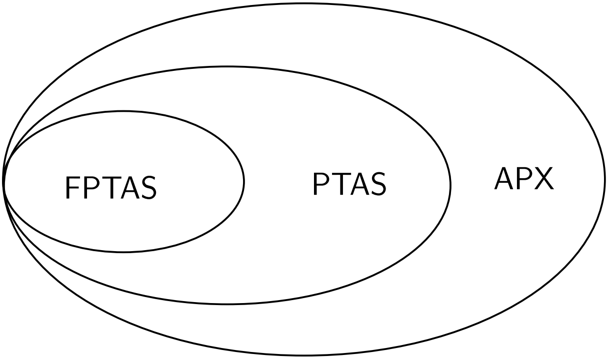
A problem is said to have a **polynomial-time approximation scheme (PTAS)** if it has a polynomial-time δ-approximation algorithm with $\delta = 1 + \epsilon$, for any fixed value $\epsilon > 0$ .The running time depends on input size and ϵ.



Approximation classes

CLASS FPTAS

A problem is said to have a **fully polynomial-time approximation scheme (or FPTAS)** if it has a PTAS with running time that is polynomial in both the input size and $1/\epsilon$.



Some theorems to prove

Let's prove that if our problem is in APX, then it will be in PTAS.

Some theorems to prove

Let's prove that if our problem is in APX, then it will be in PTAS.

Theorem

If the longest-path problem has a polynomial-time algorithm that achieves a constant factor approximation, then it has a **PTAS**.

Some theorems to prove

Let's prove that if our problem is in APX, then it will be in PTAS.

Theorem

If the longest-path problem has a polynomial-time algorithm that achieves a constant factor approximation, then it has a **PTAS**.

Then we will show that we are not really that lucky to have a PTAS.

Some theorems to prove

Let's prove that if our problem is in APX, then it will be in PTAS.

Theorem

If the longest-path problem has a polynomial-time algorithm that achieves a constant factor approximation, then it has a **PTAS**.

Then we will show that we are not really that lucky to have a PTAS.

Theorem

There is no **PTAS** for the longest-path problem, unless P = NP.

Some theorems to prove

Let's prove that if our problem is in APX, then it will be in PTAS.

Theorem

If the longest-path problem has a polynomial-time algorithm that achieves a constant factor approximation, then it has a **PTAS**.

Then we will show that we are not really that lucky to have a PTAS.

Theorem

There is no **PTAS** for the longest-path problem, unless $\mathrm{P} = \mathrm{NP}$.

Above two theorems will bring us straight to the following corollary.

Some theorems to prove

Let's prove that if our problem is in APX, then it will be in PTAS.

Theorem

If the longest-path problem has a polynomial-time algorithm that achieves a constant factor approximation, then it has a **PTAS**.

Then we will show that we are not really that lucky to have a PTAS.

Theorem

There is no **PTAS** for the longest-path problem, unless $\mathrm{P} = \mathrm{NP}$.

Above two theorems will bring us straight to the following corollary.

Corollary

There does not exist a constant factor approximation algorithm for the longest path problem, unless $\mathrm{P} = \mathrm{NP}$.

Helping Lemma

But to prove our first theorem we need to prove a lemma.

Helping Lemma

But to prove our first theorem we need to prove a lemma.

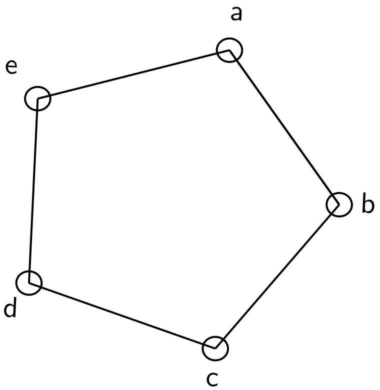
Lemma

Let $G = (V, E)$ be any graph. If the longest simple path in G has length l then the longest simple path in G^2 has length at least l^2. Moreover, given a path of length m in G^2, we can obtain in polynomial time a path of length $\sqrt{m}$ - 2 in G.

Some Definitions

Edge Square Graph

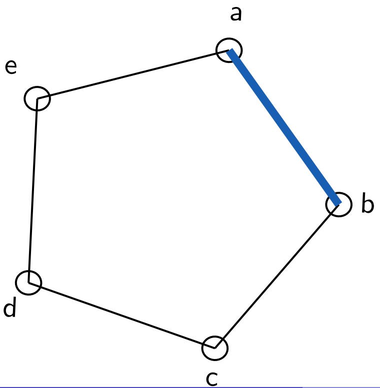
For a graph $G(V, E)$, its edge square graph $G^2(V^2, E^2)$ is obtained as follows. Replace each edge $e = (u, v)$ in G by a copy of G, call it G_e, and connect both u and v to each vertex in G_e. The vertices u and v are referred to as the **contact vertices** of G_e.

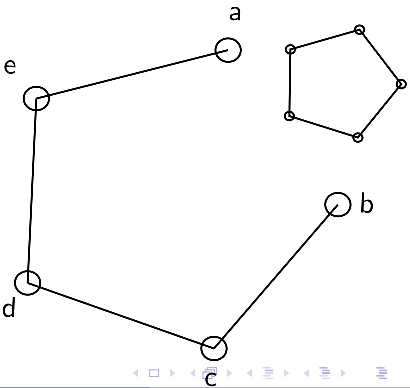


Some Definitions

Edge Square Graph

For a graph $G(V, E)$, its edge square graph $G^2(V^2, E^2)$ is obtained as follows. Replace each edge $e = (u, v)$ in G by a copy of G, call it G_e, and connect both u and v to each vertex in G_e. The vertices u and v are referred to as the **contact vertices** of G_e.

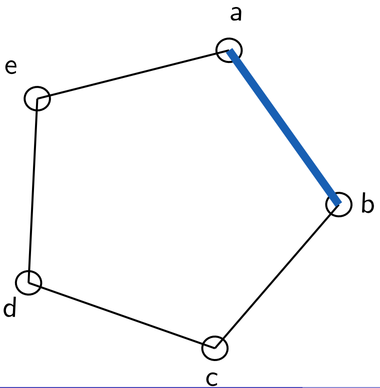


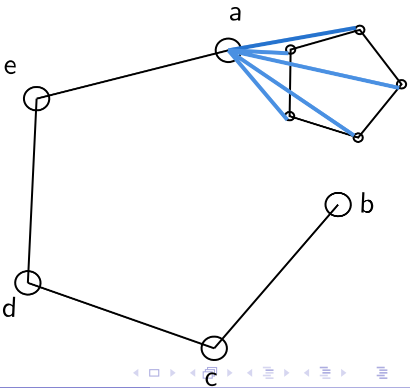


Some Definitions

Edge Square Graph

For a graph $G(V, E)$, its edge square graph $G^2(V^2, E^2)$ is obtained as follows. Replace each edge $e = (u, v)$ in G by a copy of G, call it G_e, and connect both u and v to each vertex in G_e. The vertices u and v are referred to as the **contact vertices** of G_e.

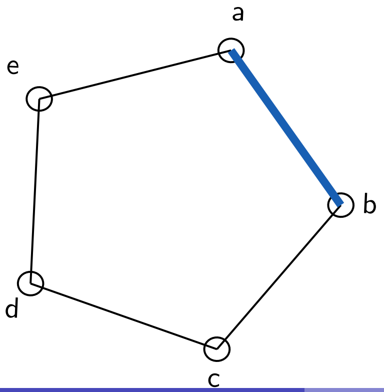


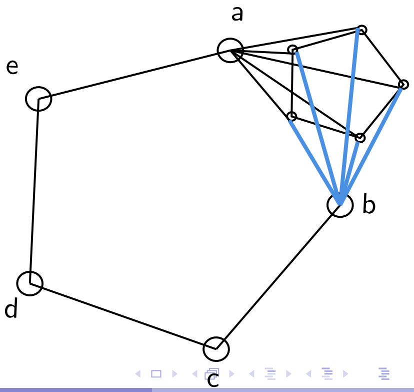


Some Definitions

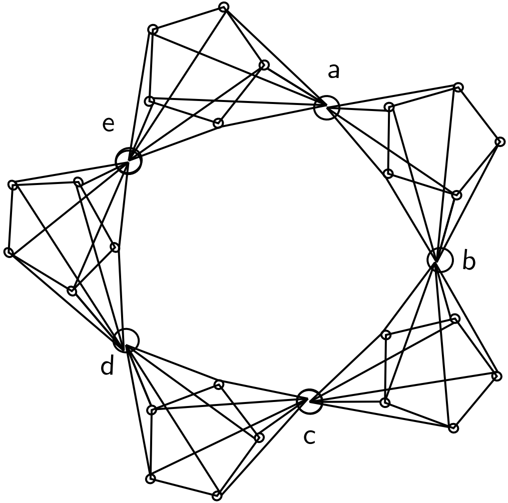
Edge Square Graph

For a graph $G(V, E)$, its edge square graph $G^2(V^2, E^2)$ is obtained as follows. Replace each edge $e = (u, v)$ in G by a copy of G, call it G_e, and connect both u and v to each vertex in G_e. The vertices u and v are referred to as the **contact vertices** of G_e.





Edge squared graph



Proof of Helping Lemma

Lemma

Let $G = (V, E)$ be any graph. **If the longest simple path in G has length l then the longest simple path in G^2 has length at least l^2.** Moreover, given a path of length m in G^2, we can obtain in polynomial time a path of length $\sqrt{m}$ - 2 in G.

Proof of Helping Lemma

Lemma

Let $G = (V, E)$ be any graph. **If the longest simple path in G has length l then the longest simple path in G^2 has length at least l^2.** Moreover, given a path of length m in G^2, we can obtain in polynomial time a path of length $\sqrt{m}$ - 2 in G.

- Say, there is a path of length l between vertices x and y in graph G. We call it path P.

Proof of Helping Lemma

Lemma

Let $G = (V, E)$ be any graph. **If the longest simple path in G has length l then the longest simple path in G^2 has length at least l^2.** Moreover, given a path of length m in G^2, we can obtain in polynomial time a path of length $\sqrt{m}$ - 2 in G.

- Say, there is a path of length l between vertices x and y in graph G. We call it path P.
- We exhibit a path P^2 in G^2.

Proof of Helping Lemma

Lemma

Let $G = (V, E)$ be any graph. **If the longest simple path in G has length l then the longest simple path in G^2 has length at least l^2.** Moreover, given a path of length m in G^2, we can obtain in polynomial time a path of length $\sqrt{m}$ - 2 in G.

- Say, there is a path of length l between vertices x and y in graph G. We call it path P.
- We exhibit a path P^2 in G^2.
- The path P^2 traverses the edge copies G_e in exactly the same order as the edges in P.

Proof of Helping Lemma

Lemma

Let $G = (V, E)$ be any graph. **If the longest simple path in G has length l then the longest simple path in G^2 has length at least l^2.** Moreover, given a path of length m in G^2, we can obtain in polynomial time a path of length $\sqrt{m}$ - 2 in G.

- Say, there is a path of length l between vertices x and y in graph G. We call it path P.
- We exhibit a path P^2 in G^2.
- The path P^2 traverses the edge copies G_e in exactly the same order as the edges in P.
- Then the length of the path P^2 is $l * (l + 2)$

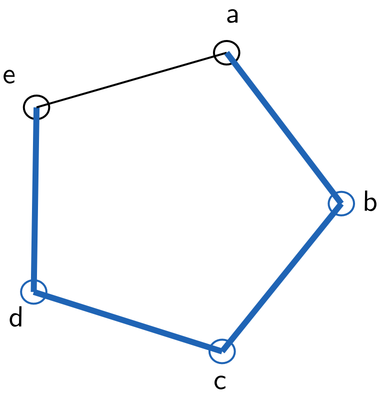
Proof of Helping Lemma

Lemma

Let $G = (V, E)$ be any graph. **If the longest simple path in G has length l then the longest simple path in G^2 has length at least l^2.** Moreover, given a path of length m in G^2, we can obtain in polynomial time a path of length $\sqrt{m}$ - 2 in G.

- Say, there is a path of length l between vertices x and y in graph G. We call it path P.
- We exhibit a path P^2 in G^2.
- The path P^2 traverses the edge copies G_e in exactly the same order as the edges in P.
- Then the length of the path P^2 is $l * (l + 2)$
- Let's see an example to brush up the concept.

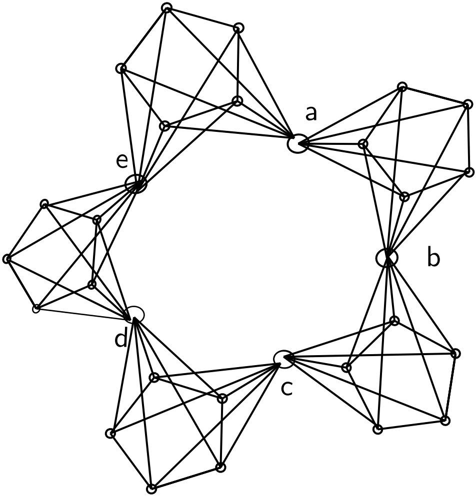
Visualization



- Graph G is shown in the figure
- It show a longest path P of length l, where $l = 4$.
- The path P visits the vertices in the following order: a, b, c, d, e.

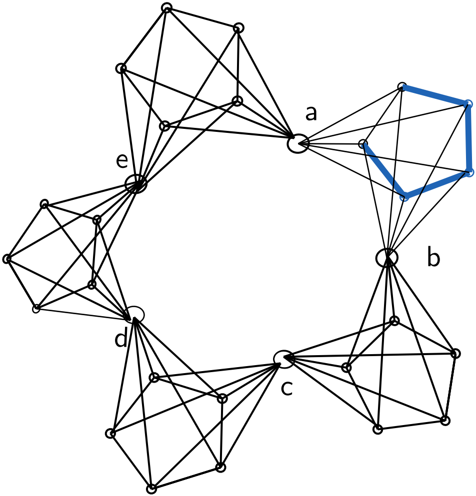
Visualization

- Graph G^2 is shown in the figure
- It exhibits the path P^2.
- P^2 traverse edge copies G_e in the exact same order as path P in graph G.
- As the path P goes in order: a, b, c, d, e path P^2 traverses the edge copies in exact same order.



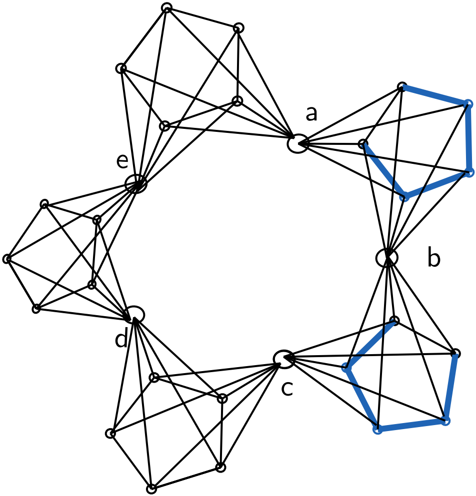
Visualization

- Graph G^2 is shown in the figure
- It exhibits the path P^2.
- P^2 traverse edge copies G_e in the exact same order as path P in graph G.
- As the path P goes in order: a, b, c, d, e path P^2 traverses the edge copies in exact same order.



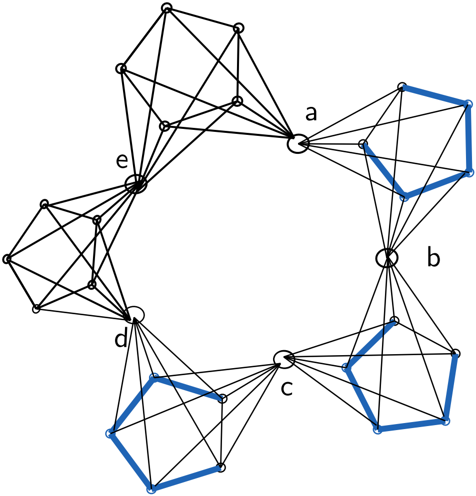
Visualization

- Graph G^2 is shown in the figure
- It exhibits the path P^2.
- P^2 traverse edge copies G_e in the exact same order as path P in graph G.
- As the path P goes in order: a, b, c, d, e path P^2 traverses the edge copies in exact same order.



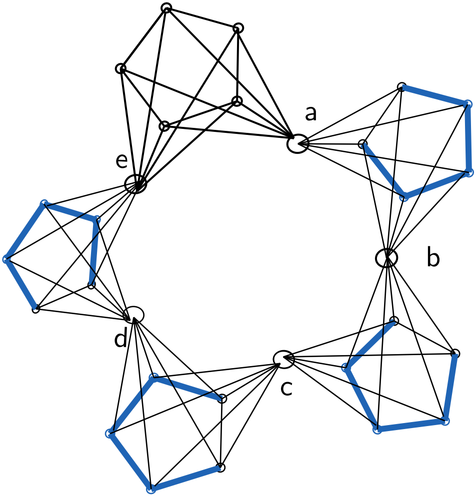
Visualization

- Graph G^2 is shown in the figure
- It exhibits the path P^2.
- P^2 traverse edge copies G_e in the exact same order as path P in graph G.
- As the path P goes in order: a, b, c, d, e path P^2 traverses the edge copies in exact same order.



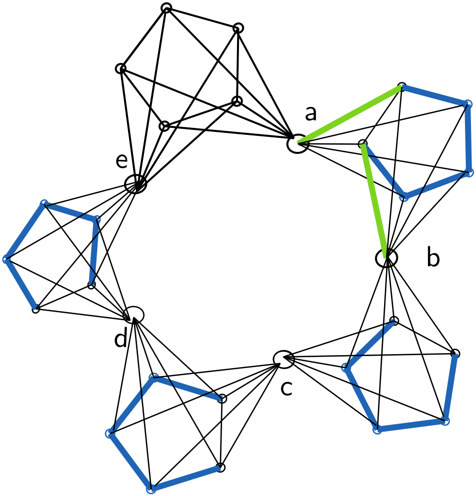
Visualization

- Graph G^2 is shown in the figure
- It exhibits the path P^2.
- P^2 traverse edge copies G_e in the exact same order as path P in graph G.
- As the path P goes in order: a, b, c, d, e path P^2 traverses the edge copies in exact same order.



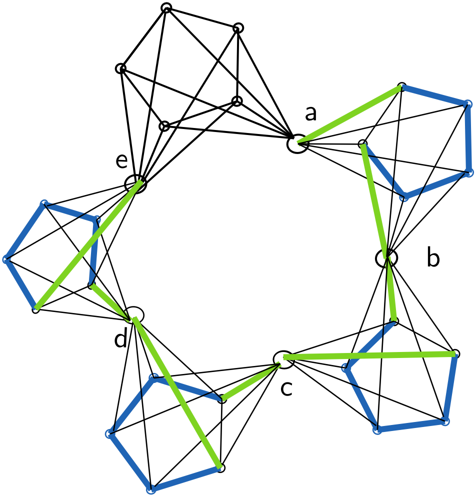
Visualization

- As the path P goes in order: a, b, c, d, e path P^2 traverses the edge copies in exact same order.
- To complete the path, the path in each edge copy G_e should be connected to the contact vertices.



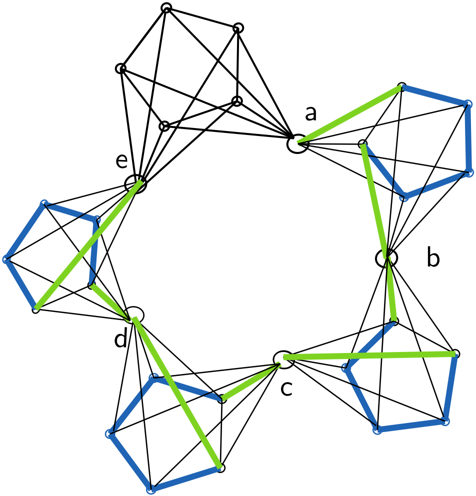
Visualization

- As the path P goes in order: a, b, c, d, e path P^2 traverses the edge copies in exact same order.
- To complete the path, the path in each edge copy G_e should be connected to the contact vertices.
- So for each edge in P, a path of length l is taken in the corresponding edge copy. And 2 edges per edge in G are taken in G^2.



Visualization

- As the path P goes in order: a, b, c, d, e path P^2 traverses the edge copies in exact same order.
- To complete the path, the path in each edge copy G_e should be connected to the contact vertices.
- So for each edge in P, a path of length l is taken in the corresponding edge copy. And 2 edges per edge in G are taken in G^2.
- Thus the total length of path P^2 is $l * (l + 2)$.



Proof of Helping Lemma (Second part)

Helping Lemma

Let $G = (V, E)$ be any graph. If the longest simple path in G has length l then the longest simple path in G^2 has length at least l^2. **Moreover, given a path of length m in G^2, we can obtain in polynomial time a path of length $\sqrt{m}$ - 2 in G.**

Proof of Helping Lemma (Second part)

Helping Lemma

Let $G = (V, E)$ be any graph. If the longest simple path in G has length l then the longest simple path in G^2 has length at least l^2. **Moreover, given a path of length m in G^2, we can obtain in polynomial time a path of length $\sqrt{m}$ - 2 in G.**

- Say, there is a path Q of length m in graph G^2.

Proof of Helping Lemma (Second part)

Helping Lemma

Let $G = (V, E)$ be any graph. If the longest simple path in G has length l then the longest simple path in G^2 has length at least l^2. **Moreover, given a path of length m in G^2, we can obtain in polynomial time a path of length $\sqrt{m}$ - 2 in G.**

- Say, there is a path Q of length m in graph G^2.
- The path Q can enter or leave a given copy of G in G^2 only via its contact vertices.

Proof of Helping Lemma (Second part)

Helping Lemma

Let $G = (V, E)$ be any graph. If the longest simple path in G has length l then the longest simple path in G^2 has length at least l^2. **Moreover, given a path of length m in G^2, we can obtain in polynomial time a path of length $\sqrt{m}$ - 2 in G.**

- Say, there is a path Q of length m in graph G^2.
- The path Q can enter or leave a given copy of G in G^2 only via its contact vertices.
- It starts off in some copy G_e, and then visits a sequence of copies that are all distinct, except that it could finish off back inside the copy G_e.

Proof of Helping Lemma (Second part)

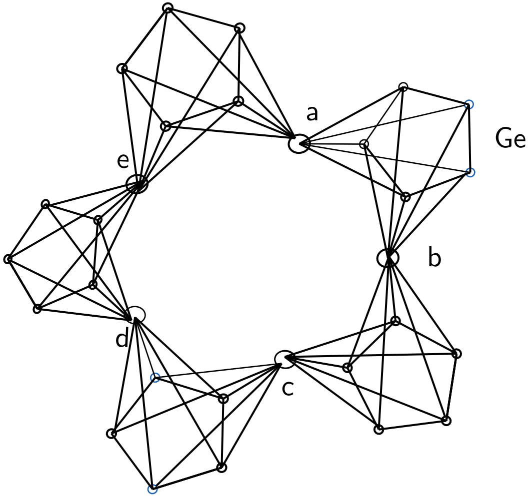
Helping Lemma

Let $G = (V, E)$ be any graph. If the longest simple path in G has length l then the longest simple path in G^2 has length at least l^2. **Moreover, given a path of length m in G^2, we can obtain in polynomial time a path of length $\sqrt{m}$ - 2 in G.**

- Say, there is a path Q of length m in graph G^2.
- The path Q can enter or leave a given copy of G in G^2 only via its contact vertices.
- It starts off in some copy G_e, and then visits a sequence of copies that are all distinct, except that it could finish off back inside the copy G_e.
- The sequence of vertices visited inside any particular edge copy forms a simple path in G, except that in G_e there could be two disjoint simple paths.

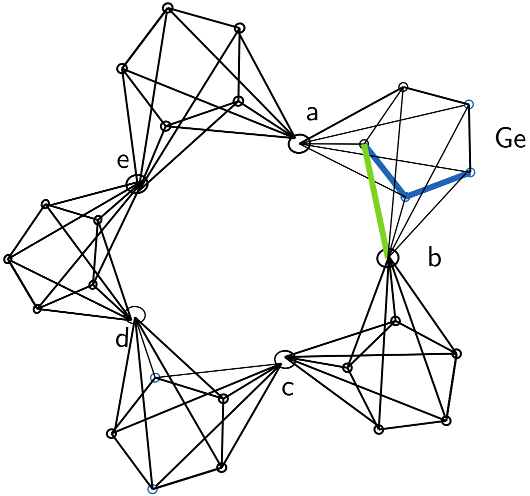
Visualization

- Graph G^2 is shown in the figure.
- We will start marking the path Q in G_2.



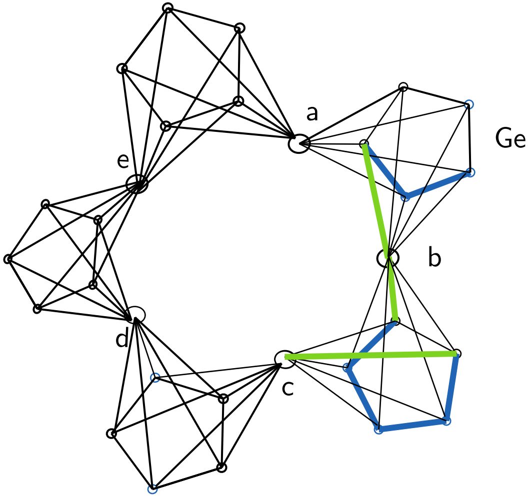
Visualization

- Graph G^2 is shown in the figure.
- We will start marking the path Q in G_2.
- The path starts off in some copy G_e.



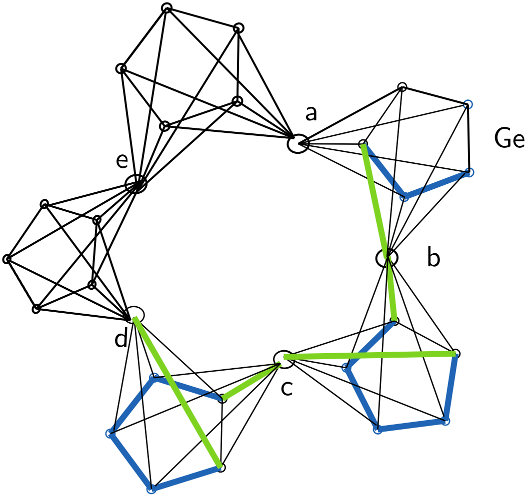
Visualization

- Graph G^2 is shown in the figure.
- We will start marking the path Q in G_2.
- The path starts off in some copy G_e.



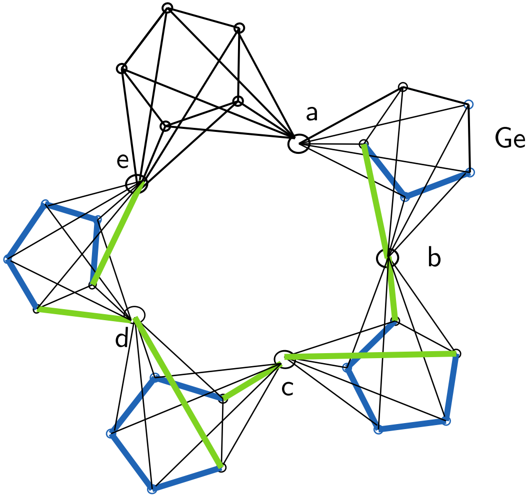
Visualization

- Graph G^2 is shown in the figure.
- We will start marking the path Q in G_2.
- The path starts off in some copy G_e.



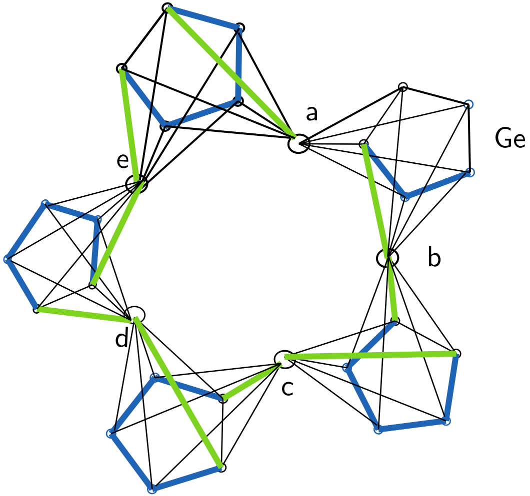
Visualization

- Graph G^2 is shown in the figure.
- We will start marking the path Q in G_2.
- The path starts off in some copy G_e.



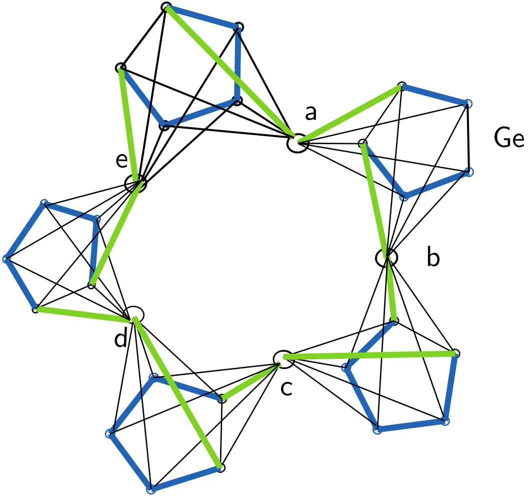
Visualization

- Graph G^2 is shown in the figure.
- We will start marking the path Q in G_2.
- The path starts off in some copy G_e.



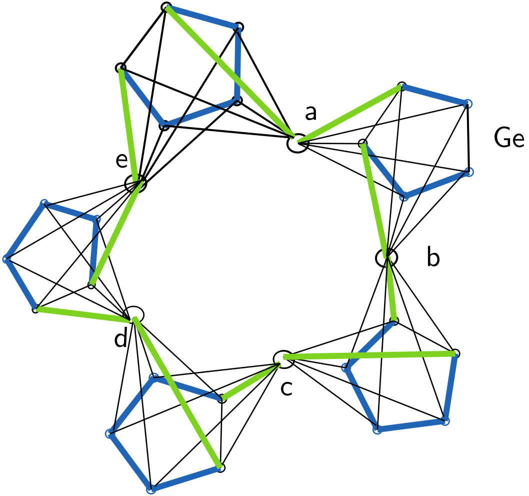
Visualization

- Graph G^2 is shown in the figure.
- We will start marking the path Q in G_2.
- The path starts off in some copy G_e.
- And it finishes back in G_e.



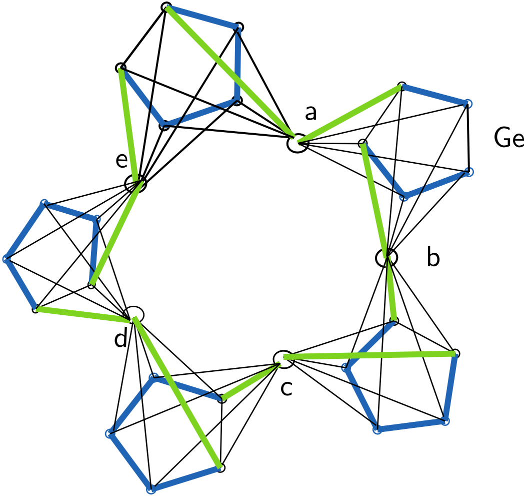
Visualization

- Graph G^2 is shown in the figure.
- We will start marking the path Q in G_2.
- The path starts off in some copy G_e.
- And it finishes back in G_e.
- All the edge copies in G^2 have one path except for G_e.



Visualization

- Graph G^2 is shown in the figure.
- We will start marking the path Q in G_2.
- The path starts off in some copy G_e.
- And it finishes back in G_e.
- All the edge copies in G^2 have one path except for G_e.
- There could be two simple paths in G_e.



Claim

We claim that one of the following holds:
Claim 1:

- One of the simple paths inside the edge copies visited by Q is of length at least $\sqrt{m} - 2$.

Claim

We claim that one of the following holds:
Claim 1:

- One of the simple paths inside the edge copies visited by Q is of length at least $\sqrt{m} - 2$.
- Let s be the length of the longest path traversed in any edge copy.

Claim

We claim that one of the following holds:
Claim 1:

- One of the simple paths inside the edge copies visited by Q is of length at least $\sqrt{m} - 2$.
- Let s be the length of the longest path traversed in any edge copy.
- So $s <= \sqrt{m} - 2$.

Claim

We claim that one of the following holds:
Claim 1:

- One of the simple paths inside the edge copies visited by Q is of length at least $\sqrt{m} - 2$.
- Let s be the length of the longest path traversed in any edge copy.
- So $s <= \sqrt{m} - 2$.

Claim 2:

Claim

We claim that one of the following holds:

Claim 1:

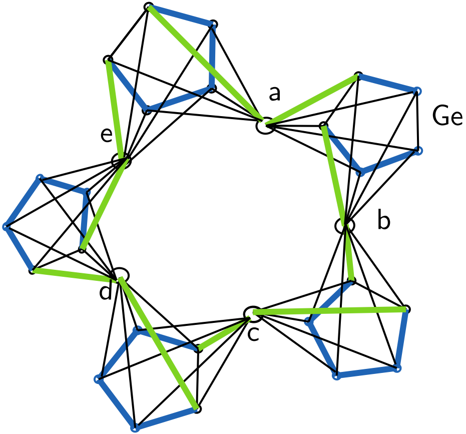
- One of the simple paths inside the edge copies visited by Q is of length at least $\sqrt{m} - 2$.
- Let s be the length of the longest path traversed in any edge copy.
- So $s <= \sqrt{m} - 2$.

Claim 2:

- Let r be the number of distinct copies of G visited. $r <= \sqrt{m} - 2$.

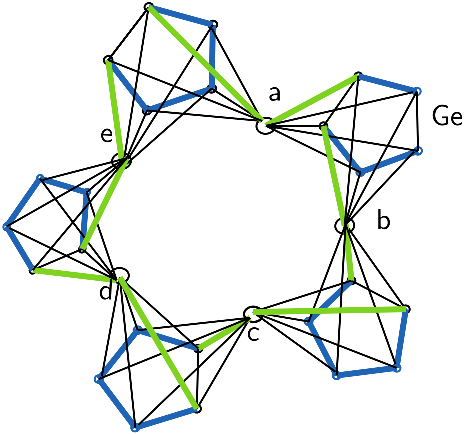
Estimating on Path Length

- Since the path Q of length m starts off and finishes back in G_e so the **total number of edge copies visited is** $(r+1)$.



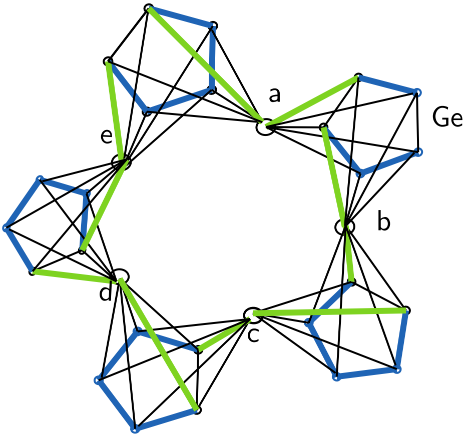
Estimating on Path Length

- Since the path Q of length m starts off and finishes back in G_e so the **total number of edge copies visited is** $(r+1)$.
- Since we have already defined longest path in any edge copy to be s, so **the length of blue-marked path is** $(r+1)*s$.



Estimating on Path Length

- Since the path Q of length m starts off and finishes back in G_e so the **total number of edge copies visited is** $(r+1)$.
- Since we have already defined longest path in any edge copy to be s, so **the length of blue-marked path is** $(r+1)*s$.
- Including the green edges which are necessary to complete the path, we have **the total length of at most** $(r+1)*s+2*r$.



Proof by Contradiction

So, what we get from our previous slide:

Length of Q, m

$m <= (r + 1) * s + 2 * r$

Proof by Contradiction

So, what we get from our previous slide:

Length of Q, m

$m <= (r + 1) * s + 2 * r$

We assume both our previous claims are violated.

Proof by Contradiction

So, what we get from our previous slide:

Length of Q, m

$m <= (r+1) * s + 2 * r$

We assume both our previous claims are violated.

Number of distint edge copies visited, r

$r < (\sqrt{m} - 2)$

Proof by Contradiction

So, what we get from our previous slide:

Length of Q, m

$m <= (r + 1) * s + 2 * r$

We assume both our previous claims are violated.

Number of distint edge copies visited, r

$r < (\sqrt{m} - 2)$

Length of longest path in any edge copy, s

$s < (\sqrt{m} - 2)$

Proof by Contradiction

Putting values of r and s in m

$m <= (r * s + s + 2 * r)$

Proof by Contradiction

Putting values of r and s in m

$m <= (r * s + s + 2 * r)$
$=> m < (\sqrt{m} - 2) * (\sqrt{m} - 2) + (\sqrt{m} - 2) + 2 * (\sqrt{m} - 2)$

Proof by Contradiction

Putting values of r and s in m

$$m <= (r * s + s + 2 * r)$$
$$=> m < (\sqrt{m} - 2) * (\sqrt{m} - 2) + (\sqrt{m} - 2) + 2 * (\sqrt{m} - 2)$$
$$=> m < (m - \sqrt{m} - 2)$$

But this is not possible.

Proof by Contradiction

Putting values of r and s in m

$m <= (r * s + s + 2 * r)$
$=> m < (\sqrt{m} - 2) * (\sqrt{m} - 2) + (\sqrt{m} - 2) + 2 * (\sqrt{m} - 2)$
$=> m < (m - \sqrt{m} - 2)$

But this is not possible.
Clearly it is a contradiction.

Proof by Contradiction

Putting values of r and s in m

$m <= (r * s + s + 2 * r)$
$=> m < (\sqrt{m} - 2) * (\sqrt{m} - 2) + (\sqrt{m} - 2) + 2 * (\sqrt{m} - 2)$
$=> m < (m - \sqrt{m} - 2)$

But this is not possible.
Clearly it is a contradiction.

So, we can safely state that,

Given the path Q in G^2, it is fairly easy to construct a path of length s in G, and also a path of length $r - 1$. This gives the desired result.

Defining APX & PTAS relationship in Longest Path

- Time to prove that if our problem is in APX, then it will be in PTAS.

Defining APX & PTAS relationship in Longest Path

- Time to prove that if our problem is in APX, then it will be in PTAS.

Theorem 1

If the longest-path problem has a polynomial-time algorithm that achieves a constant factor approximation, then it has a **PTAS**.

Defining APX & PTAS relationship in Longest Path

- Time to prove that if our problem is in APX, then it will be in PTAS.

Theorem 1

If the longest-path problem has a polynomial-time algorithm that achieves a constant factor approximation, then it has a **PTAS**.

- We will use the Helping Lemma's results for this proof.

Proving APX & PTAS relationship in Longest Path

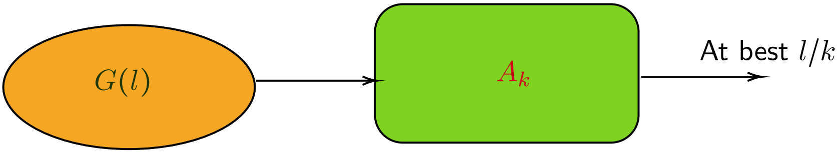
- Let A_k be a **constant factor approximation algorithm** with a performance ratio of k $(k > 1)$.

Proving APX & PTAS relationship in Longest Path

- Let A_k be a **constant factor approximation algorithm** with a performance ratio of k $(k > 1)$.
- On input graph G with longest path length l, A_k will return at best l/k length path.

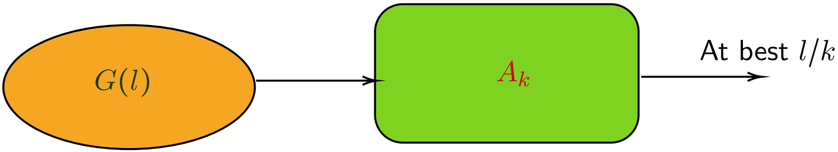
Proving APX & PTAS relationship in Longest Path

- Let A_k be a **constant factor approximation algorithm** with a performance ratio of k $(k > 1)$.
- On input graph G with longest path length l, A_k will return at best $^l/_k$ length path.



Proving APX & PTAS relationship in Longest Path

- Let A_k be a **constant factor approximation algorithm** with a performance ratio of k $(k > 1)$.
- On input graph G with longest path length l, A_k will return at best $^l/_k$ length path.



- We want to prove that this A_k should also be a **PTAS** i.e. the longest path given by A_k should be **at least** of length $\dfrac{l}{1+\epsilon}$

Proving APX & PTAS relationship in Longest Path (Cont.)

- Let p be the smallest **integer** exceeding $\log \dfrac{2\log k}{\log(1+\epsilon)}$

Proving APX & PTAS relationship in Longest Path (Cont.)

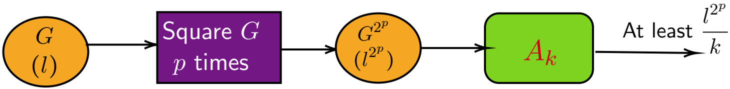
- Let p be the smallest **integer** exceeding $\log \dfrac{2 \log k}{\log(1+\epsilon)}$
- Obtain G^{2^p} by **squaring** G repeatedly **p times**.

Proving APX & PTAS relationship in Longest Path (Cont.)

- Let p be the smallest **integer** exceeding $\log \dfrac{2\log k}{\log(1+\epsilon)}$
- Obtain G^{2^p} by **squaring** G repeatedly **p times**.
- Run the approximation algorithm A_k on this G^{2^p} graph.

Proving APX & PTAS relationship in Longest Path (Cont.)

- Let p be the smallest **integer** exceeding $\log \dfrac{2\log k}{\log(1+\epsilon)}$
- Obtain G^{2^p} by **squaring** G repeatedly **p times**.
- Run the approximation algorithm A_k on this G^{2^p} graph.
- To get path length at least $\dfrac{l^{2^p}}{k}$.



Proving APX & PTAS relationship in Longest Path (Cont.)

- Using **Helping Lemma**, if A_k on G^{2^p} yields at least l^{2^p}/k, then we can obtain in G, at least $\left(\frac{l^{2^p}}{k}\right)^{\frac{1}{2^p}} - 2p$ using the following conditions.

Proving APX & PTAS relationship in Longest Path (Cont.)

- Using **Helping Lemma**, if A_k on G^{2^p} yields at least l^{2^p}/k, then we can obtain in G, at least $\left(\frac{l^{2^p}}{k}\right)^{\frac{1}{2^p}} - 2p$ using the following conditions.
- Condition I: $p \geq \log \frac{2 \log k}{\log(1+\epsilon)}$ (already stated before)
- Condition II: $l \geq \frac{4p(1+\epsilon)}{\epsilon}$

Proving APX & PTAS relationship in Longest Path (Cont.)

- Using **Helping Lemma**, if A_k on G^{2^p} yields at least l^{2^p}/k, then we can obtain in G, at least $\left(\frac{l^{2^p}}{k}\right)^{\frac{1}{2^p}} - 2p$ using the following conditions.
- Condition I: $p \geq \log \dfrac{2 \log k}{\log(1+\epsilon)}$ (already stated before)
- Condition II: $l \geq \dfrac{4p(1+\epsilon)}{\epsilon}$
- We will go from $\left(\frac{l^{2^p}}{k}\right)^{\frac{1}{2^p}} - 2p$ to $\dfrac{l}{1+\epsilon}$

Proving APX & PTAS relationship in Longest Path (Cont.)

- Using **Helping Lemma**, if A_k on G^{2^p} yields at least l^{2^p}/k, then we can obtain in G, at least $\left(\frac{l^{2^p}}{k}\right)^{\frac{1}{2^p}} - 2p$ using the following conditions.
- Condition I: $p \geq \log \frac{2 \log k}{\log(1+\epsilon)}$ (already stated before)
- Condition II: $l \geq \frac{4p(1+\epsilon)}{\epsilon}$
- We will go from $\left(\frac{l^{2^p}}{k}\right)^{\frac{1}{2^p}} - 2p$ to $\frac{l}{1+\epsilon}$
- We show that $\left(\frac{l^{2^p}}{k}\right)^{\frac{1}{2^p}} - 2p \geq \frac{l}{\sqrt{1+\epsilon}} - 2p \geq \frac{l}{1+\epsilon}$

Mathematical Manipulations - I

- Condition I was $p \geq \log \dfrac{2\log(k)}{\log(1+\epsilon)}$

Mathematical Manipulations - I

- Condition I was $p \geq \log \dfrac{2\log(k)}{\log(1+\epsilon)}$
- $2^p \geq \dfrac{2\log(k)}{\log(1+\epsilon)}$ *(Take power of 2)*

Mathematical Manipulations - I

- Condition I was $p \geq \log \dfrac{2\log(k)}{\log(1+\epsilon)}$
- $2^p \geq \dfrac{2\log(k)}{\log(1+\epsilon)}$ *(Take power of 2)*
- $\dfrac{1}{2}\log(1+\epsilon) \geq \dfrac{1}{2^p}\log(k)$ *(Re-arrange and take log)*

Mathematical Manipulations - I

- Condition I was $p \geq \log \dfrac{2\log(k)}{\log(1+\epsilon)}$
- $2^p \geq \dfrac{2\log(k)}{\log(1+\epsilon)}$ *(Take power of 2)*
- $\dfrac{1}{2}\log(1+\epsilon) \geq \dfrac{1}{2^p}\log(k)$ *(Re-arrange and take log)*
- $\log(l) - \dfrac{1}{2^p}\log(k) \geq \log(l) - \dfrac{1}{2}\log(1+\epsilon)$
  (Switch sides and subtract from $\log(l)$)

Mathematical Manipulations - I

- Condition I was $p \geq \log \dfrac{2\log(k)}{\log(1+\epsilon)}$
- $2^p \geq \dfrac{2\log(k)}{\log(1+\epsilon)}$ *(Take power of 2)*
- $\dfrac{1}{2}\log(1+\epsilon) \geq \dfrac{1}{2^p}\log(k)$ *(Re-arrange and take log)*
- $\log(l) - \dfrac{1}{2^p}\log(k) \geq \log(l) - \dfrac{1}{2}\log(1+\epsilon)$ *(Switch sides and subtract from $\log(l)$)*
- $\dfrac{1}{2^p}\left[\log\left(\dfrac{l^{2^p}}{k}\right)\right] \geq \log\left[\dfrac{l}{\sqrt{1+\epsilon}}\right]$ *(Re-arrange using log properties)*

Mathematical Manipulations - I

- Condition I was $p \geq \log \dfrac{2\log(k)}{\log(1+\epsilon)}$
- $2^p \geq \dfrac{2\log(k)}{\log(1+\epsilon)}$ *(Take power of 2)*
- $\dfrac{1}{2}\log(1+\epsilon) \geq \dfrac{1}{2^p}\log(k)$ *(Re-arrange and take log)*
- $\log(l) - \dfrac{1}{2^p}\log(k) \geq \log(l) - \dfrac{1}{2}\log(1+\epsilon)$ *(Switch sides and subtract from $\log(l)$)*
- $\dfrac{1}{2^p}\left[\log\left(\dfrac{l^{2^p}}{k}\right)\right] \geq \log\left[\dfrac{l}{\sqrt{1+\epsilon}}\right]$ *(Re-arrange using log properties)*
- $\left(\dfrac{l^{2^p}}{k}\right)^{\frac{1}{2^p}} - 2p \quad \geq \dfrac{l}{\sqrt{1+\epsilon}} - 2p$ *(Take power of $\dfrac{1}{2^p}$ and subtract $2p$)*

Mathematical Manipulations - II

- Condition II was $l \geq \dfrac{4p(1+\epsilon)}{\epsilon}$
- $-2p \geq -\dfrac{l\epsilon}{2(1+\epsilon)}$ *(Re-arrange)*

Mathematical Manipulations - II

- Condition II was $l \geq \dfrac{4p(1+\epsilon)}{\epsilon}$
- $-2p \geq -\dfrac{l\epsilon}{2(1+\epsilon)}$ *(Re-arrange)*
- $\dfrac{l}{\sqrt{1+\epsilon}} - 2p \geq \dfrac{l}{\sqrt{1+\epsilon}} - \dfrac{l\epsilon}{2(1+\epsilon)}$ *(Intermediate inequality)*

Mathematical Manipulations - II

- Condition II was $l \geq \dfrac{4p(1+\epsilon)}{\epsilon}$
- $-2p \geq -\dfrac{l\epsilon}{2(1+\epsilon)}$ *(Re-arrange)*
- $\dfrac{l}{\sqrt{1+\epsilon}} - 2p \geq \dfrac{l}{\sqrt{1+\epsilon}} - \dfrac{l\epsilon}{2(1+\epsilon)}$ *(Intermediate inequality)*

  $= l\left\{\dfrac{2\sqrt{1+\epsilon}-\epsilon}{2(1+\epsilon)}\right\}$ *(Expand left hand side)*

Mathematical Manipulations - II

- Condition II was $l \geq \dfrac{4p(1+\epsilon)}{\epsilon}$
- $-2p \geq -\dfrac{l\epsilon}{2(1+\epsilon)}$ *(Re-arrange)*
- $\dfrac{l}{\sqrt{1+\epsilon}} - 2p \geq \dfrac{l}{\sqrt{1+\epsilon}} - \dfrac{l\epsilon}{2(1+\epsilon)}$ *(Intermediate inequality)*

$$= l\left\{\frac{2\sqrt{1+\epsilon}-\epsilon}{2(1+\epsilon)}\right\}$$ *(Expand left hand side)*

$$= l\left\{\frac{2(1+\epsilon/2-\epsilon^2/4+\epsilon^3/16+\ldots)-\epsilon}{2(1+\epsilon)}\right\}$$

Mathematical Manipulations - II

- Condition II was $l \geq \dfrac{4p(1+\epsilon)}{\epsilon}$
- $-2p \geq -\dfrac{l\epsilon}{2(1+\epsilon)}$ *(Re-arrange)*
- $\dfrac{l}{\sqrt{1+\epsilon}} - 2p \geq \dfrac{l}{\sqrt{1+\epsilon}} - \dfrac{l\epsilon}{2(1+\epsilon)}$ *(Intermediate inequality)*

$$= l\left\{\frac{2\sqrt{1+\epsilon}-\epsilon}{2(1+\epsilon)}\right\} \quad \textit{(Expand left hand side)}$$

$$= l\left\{\frac{2(1+{}^{\epsilon}\!/\!{}_{2}-{}^{\epsilon^2}\!/\!{}_{4}+{}^{\epsilon^3}\!/\!{}_{16}+\ldots)-\epsilon}{2(1+\epsilon)}\right\}$$

$$= l\left\{\frac{2(1+{}^{\epsilon}\!/\!{}_{2})-\epsilon}{2(1+\epsilon)}\right\} \quad \textit{(Since } \epsilon < 1\textit{)}$$

Mathematical Manipulations - II

- Condition II was $l \geq \dfrac{4p(1+\epsilon)}{\epsilon}$
- $-2p \geq -\dfrac{l\epsilon}{2(1+\epsilon)}$ *(Re-arrange)*
- $\dfrac{l}{\sqrt{1+\epsilon}} - 2p \geq \dfrac{l}{\sqrt{1+\epsilon}} - \dfrac{l\epsilon}{2(1+\epsilon)}$ *(Intermediate inequality)*

$$= l\left\{\frac{2\sqrt{1+\epsilon}-\epsilon}{2(1+\epsilon)}\right\} \quad \textit{(Expand left hand side)}$$

$$= l\left\{\frac{2(1+\epsilon/2-\epsilon^2/4+\epsilon^3/16+\ldots)-\epsilon}{2(1+\epsilon)}\right\}$$

$$= l\left\{\frac{2(1+\epsilon/2)-\epsilon}{2(1+\epsilon)}\right\} \quad \textit{(Since } \epsilon < 1\textit{)}$$

$$= \frac{l}{1+\epsilon}$$

Mathematical Manipulations - II

- Condition II was $l \geq \dfrac{4p(1+\epsilon)}{\epsilon}$
- $-2p \geq -\dfrac{l\epsilon}{2(1+\epsilon)}$ *(Re-arrange)*
- $\dfrac{l}{\sqrt{1+\epsilon}} - 2p \geq \dfrac{l}{\sqrt{1+\epsilon}} - \dfrac{l\epsilon}{2(1+\epsilon)}$ *(Intermediate inequality)*

$$= l\left\{\frac{2\sqrt{1+\epsilon}-\epsilon}{2(1+\epsilon)}\right\} \quad \textit{(Expand left hand side)}$$

$$= l\left\{\frac{2(1+{}^{\epsilon}/_{2}-{}^{\epsilon^2}/_{4}+{}^{\epsilon^3}/_{16}+\ldots)-\epsilon}{2(1+\epsilon)}\right\}$$

$$= l\left\{\frac{2(1+{}^{\epsilon}/_{2})-\epsilon}{2(1+\epsilon)}\right\} \quad \textit{(Since } \epsilon < 1\textit{)}$$

$$= \frac{l}{1+\epsilon}$$

Hence, $\dfrac{l}{\sqrt{1+\epsilon}} - \dfrac{l\epsilon}{2(1+\epsilon)} \geq \dfrac{l}{1+\epsilon}$ *(Required expression for PTAS)*

Proving APX & PTAS relationship (Finishing Touch)

- We showed that A_k (constant factor approximation algorithm in k) should also be a PTAS.

Proving APX & PTAS relationship (Finishing Touch)

- We showed that A_k (constant factor approximation algorithm in k) should also be a PTAS.
- Running time of A_k is fixed for ϵ.
- As G^{2^p} has at most n^{3^p} vertices.

Proving APX & PTAS relationship (Finishing Touch)

- We showed that A_k (constant factor approximation algorithm in k) should also be a PTAS.
- Running time of A_k is fixed for ϵ.
- As G^{2^p} has at most n^{3^p} vertices.
- Next, we will show that PTAS does not exist for Longest Path problem.

Proving Longest Path has no PTAS

- Time to prove that longest path problem has no PTAS.

Theorem 2

There is no PTAS for the Longest Path problem; **unless P = NP**

- We will use results from **Papadimitriou and Yannaka (1993)** [1] and **Arora et al. (1998)** [2].

Proving Longest Path has no PTAS

- Time to prove that longest path problem has no PTAS.

Theorem 2

There is no PTAS for the Longest Path problem; **unless P = NP**

- We will use results from **Papadimitriou and Yannaka (1993)** [1] and **Arora et al. (1998)** [2].
- We will show hardness results on a restricted instance i.e. instances containing **Hamiltonian Cycle**.

Proving Longest Path has no PTAS

- Time to prove that longest path problem has no PTAS.

Theorem 2

There is no PTAS for the Longest Path problem; **unless P = NP**

- We will use results from **Papadimitriou and Yannaka (1993)** [1] and **Arora et al. (1998)** [2].
- We will show hardness results on a restricted instance i.e. instances containing **Hamiltonian Cycle**.
- If it is NP-hard to find a PTAS in this restricted instance, then for sure it will be NP-Hard on more **general case**.

Proving Longest Path has no PTAS (Cont.)

- Let CTSP (custom TSP) be the problem of finding optimal TSP tour in a complete graph where all edge lengths are **either 1 or 2**.

Proving Longest Path has no PTAS (Cont.)

- Let CTSP (custom TSP) be the problem of finding optimal TSP tour in a complete graph where all edge lengths are **either 1 or 2**.
- **Papadimitriou and Yannaka (1993)** [1] showed that the approximation version of this problem is **MAX SNP-Hard**.
- This reduction is from a version of MAX 3SAT problem which is also **MAX SNP-Complete**.

Proving Longest Path has no PTAS (Cont.)

- Let CTSP (custom TSP) be the problem of finding optimal TSP tour in a complete graph where all edge lengths are **either 1 or 2**.
- **Papadimitriou and Yannaka (1993)** [1] showed that the approximation version of this problem is **MAX SNP-Hard**.
- This reduction is from a version of MAX 3SAT problem which is also **MAX SNP-Complete**.
- **MAX SNP** problems are those that have constant factor approximation algorithms, but no approximation schemes **unless P = NP**.

Combining Longest Path & TSP

- Following from the results of **Papadimitriou and Yannaka (1993)** [1], we claim that for every $\delta > 0$, if there exists a polynomial time algorithm on which any instance of CTSP (optimal tour len $= n$) returns a tour of cost at most $(1 + \delta)n$, then MAX3SAT has a PTAS.
- Optimal tour of n is only possible with having Hamiltonian Cycle of all edges with weights 1 taken.

Combining Longest Path & TSP

- Following from the results of **Papadimitriou and Yannaka (1993)** [1], we claim that for every $\delta > 0$, if there exists a polynomial time algorithm on which any instance of CTSP (optimal tour len $= n$) returns a tour of cost at most $(1 + \delta)n$, then MAX3SAT has a PTAS.
- Optimal tour of n is only possible with having Hamiltonian Cycle of all edges with weights 1 taken.
- Suppose we have a PTAS for Longest Path in Hamiltonian Graphs, then we can find paths of length at least $(1 - \delta)n$ in them.

Combining Longest Path & TSP

- Following from the results of **Papadimitriou and Yannaka (1993)** [1], we claim that for every $\delta > 0$, if there exists a polynomial time algorithm on which any instance of CTSP (optimal tour len $= n$) returns a tour of cost at most $(1 + \delta)n$, then MAX3SAT has a PTAS.
- Optimal tour of n is only possible with having Hamiltonian Cycle of all edges with weights 1 taken.
- Suppose we have a PTAS for Longest Path in Hamiltonian Graphs, then we can find paths of length at least $(1 - \delta)n$ in them.
- Using PTAS for Longest Path, we can construct a path of length $(1 - \delta)n$ using edges weights only 1.

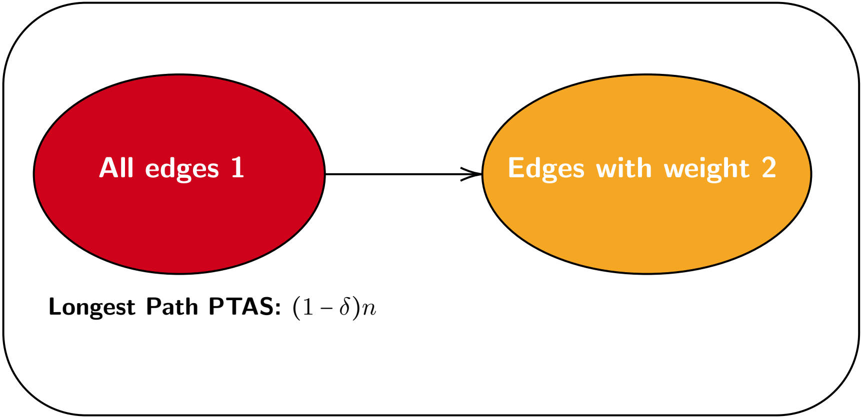
Combining Longest Path & TSP

- Following from the results of **Papadimitriou and Yannaka (1993)** [1], we claim that for every $\delta > 0$, if there exists a polynomial time algorithm on which any instance of CTSP (optimal tour len $= n$) returns a tour of cost at most $(1+\delta)n$, then MAX3SAT has a PTAS.
- Optimal tour of n is only possible with having Hamiltonian Cycle of all edges with weights 1 taken.
- Suppose we have a PTAS for Longest Path in Hamiltonian Graphs, then we can find paths of length at least $(1-\delta)n$ in them.
- Using PTAS for Longest Path, we can construct a path of length $(1-\delta)n$ using edges weights only 1.
- This can be converted into a tour of weight at most $(1+\delta)n$ in the instance of CTSP by extending the path with edge weights 2.

Combining Longest Path & TSP

- Following from the results of **Papadimitriou and Yannaka (1993)** [1], we claim that for every $\delta > 0$, if there exists a polynomial time algorithm on which any instance of CTSP (optimal tour len $= n$) returns a tour of cost at most $(1 + \delta)n$, then MAX3SAT has a PTAS.
- Optimal tour of n is only possible with having Hamiltonian Cycle of all edges with weights 1 taken.
- Suppose we have a PTAS for Longest Path in Hamiltonian Graphs, then we can find paths of length at least $(1 - \delta)n$ in them.
- Using PTAS for Longest Path, we can construct a path of length $(1 - \delta)n$ using edges weights only 1.
- This can be converted into a tour of weight at most $(1 + \delta)n$ in the instance of CTSP by extending the path with edge weights 2.
- Hence, the PTAS for Longest Path can be used to obtain PTAS for CTSP instances of the restricted class.

Visualizing PTAS construction of Longesth Path & TSP



CTSP PTAS: $(1+\delta)n$

Finalizing Proof

- PTAS for Longest Path is used to obtain PTAS for CTSP (instances of this restricted class).

Finalizing Proof

- PTAS for Longest Path is used to obtain PTAS for CTSP (instances of this restricted class).
- Following the results from **Papadimitriou and Yannaka (1993)** [1], since CTSP returns a tour of cost at most $(1+\delta)n$, then MAX3SAT must have a PTAS.

Finalizing Proof

- PTAS for Longest Path is used to obtain PTAS for CTSP (instances of this restricted class).
- Following the results from **Papadimitriou and Yannaka (1993)** [1], since CTSP returns a tour of cost at most $(1+\delta)n$, then MAX3SAT must have a PTAS.
- **Arora et al. (1998)** [2] concluded that that if any **MAX SNP-Hard problem** has a PTAS, then P $=$ NP.

Finalizing Proof

- PTAS for Longest Path is used to obtain PTAS for CTSP (instances of this restricted class).
- Following the results from **Papadimitriou and Yannaka (1993)** [1], since CTSP returns a tour of cost at most $(1 + \delta)n$, then MAX3SAT must have a PTAS.
- **Arora et al. (1998)** [2] concluded that that if any **MAX SNP-Hard problem** has a PTAS, then P $=$ NP.
- So, MAX3SAT (a MAX SNP-Complete problem) has a PTAS, which leads to P $=$ NP.
- Hence, we can conclude that Longest Path does not have a PTAS, unless P $=$ NP.

Summary of proofs (so far)

- We have proven the following theorems.

Theorem

If the longest-path problem has a polynomial-time algorithm that achieves a **constant factor approximation**, then it has a **PTAS**.

Theorem

There is **no PTAS** for the Longest Path problem; **unless P = NP**

Summary of proofs (so far)

- We have proven the following theorems.

Theorem

If the longest-path problem has a polynomial-time algorithm that achieves a **constant factor approximation**, then it has a **PTAS**.

Theorem

There is **no PTAS** for the Longest Path problem; **unless P = NP**

- These lead to the following corollary.

Corollary

There does not exist a **constant factor approximation algorithm** for the longest path problem, **unless P = NP**

Summary of proofs (so far)

- We have proven the following theorems.

Theorem

If the longest-path problem has a polynomial-time algorithm that achieves a **constant factor approximation**, then it has a **PTAS**.

Theorem

There is **no PTAS** for the Longest Path problem; **unless P = NP**

- These lead to the following corollary.

Corollary

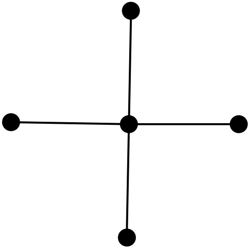
There does not exist a **constant factor approximation algorithm** for the longest path problem, **unless P = NP**

- All the proofs presented so far are from **Karger et al. (1997)** [3].

Approximation Algorithm for a Special Case

- We present a $2/3$-approximation algorithm for finding a longest path in a special case of solid grid graphs, based on the findings of A. A. Sardroud and A. Bagheri (2016). [4]

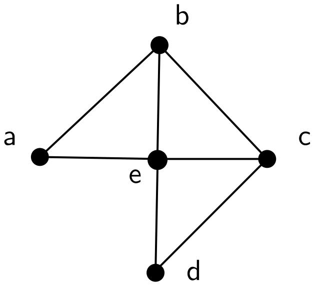
Preliminaries



- **Cut Vertex:** A vertex whose removal increases the number of connected components in a graph. Removing a cut vertex from a graph breaks it in to two or more graphs.

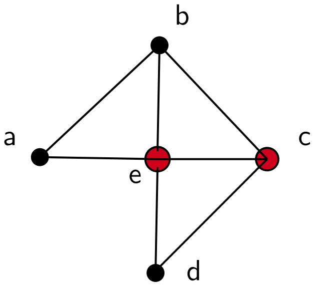
Preliminaries

- **2-Connected Graph:** A connected graph G is 2-connected if it contains no cut vertex, i.e. a vertex whose removal increases the number of connected components of G. Figure shows a 2-connected graph.

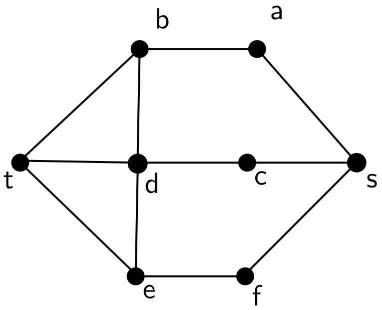


Preliminaries

- **Cutting Pair:** Two vertices of a 2-connected graph G are called cutting pair if their removal disconnects G. Removing e and c will disconnect this 2-connected graph. Cutting pair {e, c}

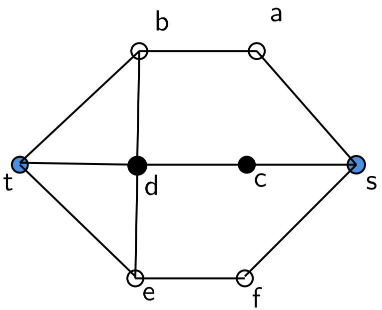


Preliminaries



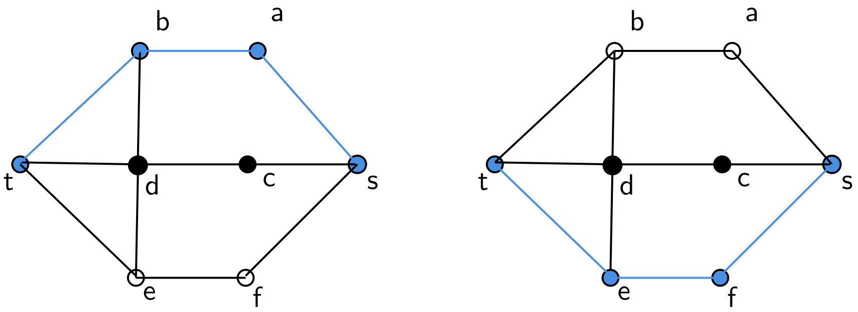
- **Boundary Vertices:** The vertices of graph G adjacent to the outer face are called boundary vertices, and the set of boundary vertices of G forms its boundary.
  Boundary vertices: {s, a, b, t, e, f}

Preliminaries



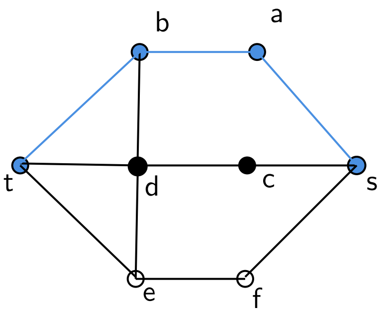
- **Boundary Path:** A path in graph G is a boundary path if it contains only the boundary vertices of G.

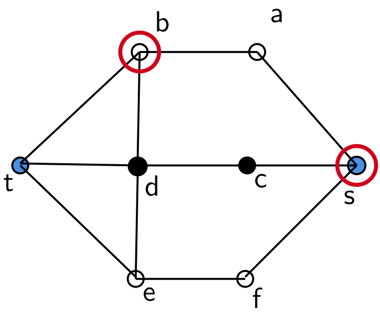
Preliminaries



- **Boundary Path:** There can be 2 boundary paths from this graph. One path contains A: {t, b, a, s} vertices and another path contains B: {t, e, f, s} vertices.

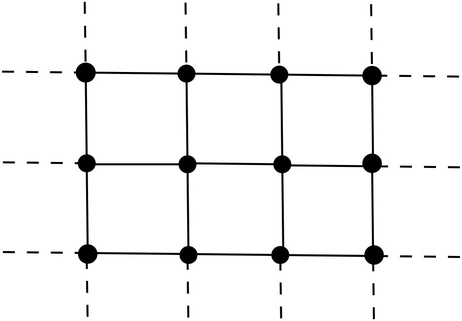
Preliminaries

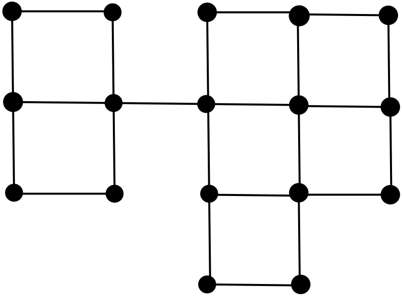




- **Critical Cutting Pair:** The vertex b on the path A: {t, b, a, s} and s to be a critical cutting pair if b and s are a cutting pair and there is no vertex u between s and b on path A such that s and u are a cutting pair. In other words, b is a critical cutting pair to s if it is the nearest vertex to t on path A which is a cutting pair to s.

Preliminaries





- **Infinite Integer Grid:** On the right side we have a graph G which is a solid grid graph, that is, a vertex-induced sub-graph of the infinite integer grid whose vertices are the integer coordinated points of the plane and has an edge between any two vertices of distance one. In addition, considering their natural embedding on the integer grid, we assume that solid grid graphs are plane graphs.

Forbidden Problems

- For a 2-connected solid grid graph G, a source boundary vertex s and destination boundary vertex t, ALP(G, s, t) is defined to be the problem of finding a path between s and t of G, of length at least $2/3$ of the length of the longest path.

Forbidden Problems

- For a 2-connected solid grid graph G, a source boundary vertex s and destination boundary vertex t, ALP(G, s, t) is defined to be the problem of finding a path between s and t of G, of length at least $2/3$ of the length of the longest path.
- The forbidden problems are the problems in which there is no path of length at least $2n/3$ between s and t in G.

Forbidden Problems

- For a 2-connected solid grid graph G, a source boundary vertex s and destination boundary vertex t, ALP(G, s, t) is defined to be the problem of finding a path between s and t of G, of length at least $2/3$ of the length of the longest path.
- The forbidden problems are the problems in which there is no path of length at least $2n/3$ between s and t in G.
- ALP(G, s, t) is forbidden if it satisfies one of the conditions **(F1)**, **(F2)**, **(F3)** or **(F4)**.

Forbidden Problems

- For a 2-connected solid grid graph G, a source boundary vertex s and destination boundary vertex t, ALP(G, s, t) is defined to be the problem of finding a path between s and t of G, of length at least $2/3$ of the length of the longest path.
- The forbidden problems are the problems in which there is no path of length at least $2n/3$ between s and t in G.
- ALP(G, s, t) is forbidden if it satisfies one of the conditions **(F1)**, **(F2)**, **(F3)** or **(F4)**.
- For defining (F1) and (F2), we should first define **(C1)** and **(C2)**.

Forbidden Problems

(C1)

ALP(G, s, t) satisfies (C1) if s and t are a cutting pair, or G has a vertex v such that s and v are a cutting pair and ALP$(G \cup \{v\}, v, t)$ satisfies (C1), in which G' is the connected component of $G \setminus \{s, v\}$ which contains t.

Forbidden Problems

(C1)

ALP(G, s, t) satisfies (C1) if s and t are a cutting pair, or G has a vertex v such that s and v are a cutting pair and ALP$(G \cup \{v\}, v, t)$ satisfies (C1), in which G' is the connected component of $G \setminus \{s, v\}$ which contains t.

- If ALP(G, s, t) satisfies (C1), there should be an integer $m \geq 1$ and a sequence of vertices $v_0 = s, v_1, \ldots, v_m = t$ such that each two consecutive vertices of the sequence are a cutting pair.

Forbidden Problems

(C1)

ALP(G, s, t) satisfies (C1) if s and t are a cutting pair, or G has a vertex v such that s and v are a cutting pair and ALP$(G \cup \{v\}, v, t)$ satisfies (C1), in which G' is the connected component of $G \smallsetminus \{s, v\}$ which contains t.

- If ALP(G, s, t) satisfies (C1), there should be an integer $m \geq 1$ and a sequence of vertices $v_0 = s, v_1, \ldots, v_m = t$ such that each two consecutive vertices of the sequence are a cutting pair.
- If an ALP(G, s, t) satisfies (C1), we define graph G_0 is the same as G, and for $0 \leq i \leq m-2$, C_i is the connected component of $G_i \smallsetminus \{v_i, v_{i+1}\}$ which does not contain t, and G_{i+1} is $G_i \smallsetminus (C_i \cup \{v_i\})$.

Forbidden Problems

(C1)

ALP(G, s, t) satisfies (C1) if s and t are a cutting pair, or G has a vertex v such that s and v are a cutting pair and ALP$(G \cup \{v\}, v, t)$ satisfies (C1), in which G' is the connected component of $G \setminus \{s, v\}$ which contains t.

- If ALP(G, s, t) satisfies (C1), there should be an integer $m \geq 1$ and a sequence of vertices $v_0 = s, v_1, \ldots, v_m = t$ such that each two consecutive vertices of the sequence are a cutting pair.
- If an ALP(G, s, t) satisfies (C1), we define graph G_0 is the same as G, and for $0 \leq i \leq m - 2$, C_i is the connected component of $G_i \setminus \{v_i, v_{i+1}\}$ which does not contain t, and G_{i+1} is $G_i \setminus (C_i \cup \{v_i\})$.

Lemma

If ALP(G, s, t) satisfies (C1), for each path between s and t in G, there is an $0 \leq i \leq m$ such that the path does not contain any vertex of C_i.

Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (C1), for each path between s and t in G, there is an $0 \leq i \leq m$ such that the path does not contain any vertex of C_i.

- Let P be a path between $s = v_0$ and t in G containing at least one vertex of each $C_i, 0 \leq i \leq m$.

Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (C1), for each path between s and t in G, there is an $0 \leq i \leq m$ such that the path does not contain any vertex of C_i.

- Let P be a path between $s = v_0$ and t in G containing at least one vertex of each $C_i, 0 \leq i \leq m$.
- Because v_0 and v_1 are a cutting pair and t is not in C_0, if P contains a vertex of C_0, it should come before v_1 along the path P, and v_1 should precede all the vertices of $G_1 \setminus v_1$ in P.

Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (C1), for each path between s and t in G, there is an $0 \leq i \leq m$ such that the path does not contain any vertex of C_i.

- Let P be a path between $s = v_0$ and t in G containing at least one vertex of each $C_i, 0 \leq i \leq m$.
- Because v_0 and v_1 are a cutting pair and t is not in C_0, if P contains a vertex of C_0, it should come before v_1 along the path P, and v_1 should precede all the vertices of $G_1 \setminus v_1$ in P.
- If P contains some vertices of C_{m-2}, they should precede v_{m-1} in P, and all the vertices of C_{m-1} and C_m should come after v_{m-1} along P.

Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (C1), for each path between s and t in G, there is an $0 \leq i \leq m$ such that the path does not contain any vertex of C_i.

- Let P be a path between $s = v_0$ and t in G containing at least one vertex of each $C_i, 0 \leq i \leq m$.
- Because v_0 and v_1 are a cutting pair and t is not in C_0, if P contains a vertex of C_0, it should come before v_1 along the path P, and v_1 should precede all the vertices of $G_1 \setminus v_1$ in P.
- If P contains some vertices of C_{m-2}, they should precede v_{m-1} in P, and all the vertices of C_{m-1} and C_m should come after v_{m-1} along P.
- Because v_{m-1} and v_m are a cutting pair, P cannot contain vertices from both of C_{m-1} and C_m, which leads to a contradiction.

Forbidden Problems

(C1)

ALP(G, s, t) satisfies (C1) if s and t are a cutting pair, or G has a vertex v such that s and v are a cutting pair and ALP$(G' \cup \{v\}, v, t)$ satisfies (C1), in which G' is the connected component of $G \setminus \{s, v\}$ which contains t.

Forbidden Problems

(C1)

ALP(G, s, t) satisfies (C1) if s and t are a cutting pair, or G has a vertex v such that s and v are a cutting pair and ALP$(G' \cup \{v\}, v, t)$ satisfies (C1), in which G' is the connected component of $G \setminus \{s, v\}$ which contains t.

(F1)

ALP(G, s, t) satisfies (F1) if it satisfies (C1) such that all graphs C_i, $0 \leq i \leq m$, contain more than one vertex.

Forbidden Problems

(C2)

ALP(G, s, t) satisfies (C2) if G contains two vertices that are a critical cutting pair to s, or G has a vertex v cutting pair to s and ALP$(G' \cup \{v\}, v, t)$ satisfies (C2), in which G' is the connected component of $G \setminus \{s, v\}$ which contains t.

Forbidden Problems

(C2)

ALP(G, s, t) satisfies (C2) if G contains two vertices that are a critical cutting pair to s, or G has a vertex v cutting pair to s and ALP$(G' \cup \{v\}, v, t)$ satisfies (C2), in which G' is the connected component of $G \setminus \{s, v\}$ which contains t.

- If ALP(G, s, t) satisfies (C2), there should be a sequence of vertices $v_0 = s, v_1, \ldots, v_m$ such that each two consecutive vertices of the sequence are a cutting pair and two vertices v_{m+1} and v_{m+2} which are both cutting pairs to v_m, for an integer m.

Forbidden Problems

(C2)

ALP(G, s, t) satisfies (C2) if G contains two vertices that are a critical cutting pair to s, or G has a vertex v cutting pair to s and ALP$(G' \cup \{v\}, v, t)$ satisfies (C2), in which G' is the connected component of $G \smallsetminus \{s, v\}$ which contains t.

- If ALP(G, s, t) satisfies (C2), there should be a sequence of vertices $v_0 = s, v_1, \ldots, v_m$ such that each two consecutive vertices of the sequence are a cutting pair and two vertices v_{m+1} and v_{m+2} which are both cutting pairs to v_m, for an integer m.
- When ALP(G, s, t) satisfies (C2), we define graph $G_0, \ldots, G_m$ and $C_0, \ldots, C_{m+1}$ as follows: Graph G_0 is the same as G and, for $0 \leq i \leq m-1$, C_i is the connected component of G_i $\{v_i, v_{i+1}\}$ which does not contain t, and G_{i+1} is $G_i \smallsetminus (C_i \cup v_i)$.

Forbidden Problems

(C2)

ALP(G, s, t) satisfies (C2) if G contains two vertices that are a critical cutting pair to s, or G has a vertex v cutting pair to s and ALP$(G' \cup \{v\}, v, t)$ satisfies (C2), in which G' is the connected component of $G \setminus \{s, v\}$ which contains t.

- If ALP(G, s, t) satisfies (C2), there should be a sequence of vertices $v_0 = s, v_1, \ldots, v_m$ such that each two consecutive vertices of the sequence are a cutting pair and two vertices v_{m+1} and v_{m+2} which are both cutting pairs to v_m, for an integer m.
- When ALP(G, s, t) satisfies (C2), we define graph $G_0, \ldots, G_m$ and $C_0, \ldots, C_{m+1}$ as follows: Graph G_0 is the same as G and, for $0 \leq i \leq m-1$, C_i is the connected component of G_i $\{v_i, v_{i+1}\}$ which does not contain t, and G_{i+1} is $G_i \setminus (C_i \cup v_i)$.
- C_m and C_{m+1} are, respectively, the connected components of $G_m \setminus \{v_m, v_{m+1}\}$ and $G_m \setminus \{v_m, v_{m+2}\}$ which do not contain t.

Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (C2), for each path between s and t in G, there is an $0 \leq i \leq m + 1$ such that the path does not contain any vertex of C_i.

- Let P be a path between $s = v_0$ and t in G containing at least one vertex of each $C_i, 0 \leq i \leq m + 1$.

Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (C2), for each path between s and t in G, there is an $0 \leq i \leq m+1$ such that the path does not contain any vertex of C_i.

- Let P be a path between $s = v_0$ and t in G containing at least one vertex of each $C_i, 0 \leq i \leq m+1$.
- We can show that if P contains at least one vertex of each C_i, $0 \leq i \leq m$, v_m should precede all the vertices of $G_m \setminus v_m$ along P.

Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (C2), for each path between s and t in G, there is an $0 \le i \le m+1$ such that the path does not contain any vertex of C_i.

- Let P be a path between $s = v_0$ and t in G containing at least one vertex of each $C_i, 0 \le i \le m+1$.
- We can show that if P contains at least one vertex of each C_i, $0 \le i \le m$, v_m should precede all the vertices of $G_m \setminus v_m$ along P.
- But, because v_m is a critical cutting pair to both v_{m+1} and v_{m+2}, P cannot pass through both of C_m and C_{m+1}, which is a contradiction.

Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (C2), for each path between s and t in G, there is an $0 \leq i \leq m+1$ such that the path does not contain any vertex of C_i.

- Let P be a path between $s = v_0$ and t in G containing at least one vertex of each $C_i, 0 \leq i \leq m+1$.
- We can show that if P contains at least one vertex of each C_i, $0 \leq i \leq m$, v_m should precede all the vertices of $G_m \setminus v_m$ along P.
- But, because v_m is a critical cutting pair to both v_{m+1} and v_{m+2}, P cannot pass through both of C_m and C_{m+1}, which is a contradiction.

(F2)

ALP(G, s, t) satisfies (F2) if it satisfies (C2) such that all graphs C_i, $0 \leq i \leq m+1$, contain more than one vertex.

Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (C2), for each path between s and t in G, there is an $0 \leq i \leq m+1$ such that the path does not contain any vertex of C_i.

- Let P be a path between $s = v_0$ and t in G containing at least one vertex of each $C_i, 0 \leq i \leq m+1$.
- We can show that if P contains at least one vertex of each C_i, $0 \leq i \leq m$, v_m should precede all the vertices of $G_m \setminus v_m$ along P.
- But, because v_m is a critical cutting pair to both v_{m+1} and v_{m+2}, P cannot pass through both of C_m and C_{m+1}, which is a contradiction.

(F2)

ALP(G, s, t) satisfies (F2) if it satisfies (C2) such that all graphs C_i, $0 \leq i \leq m+1$, contain more than one vertex.

- Both (F1) and (F2) may hold in a problem ALP(G, s, t) simultaneously.

Forbidden Problems

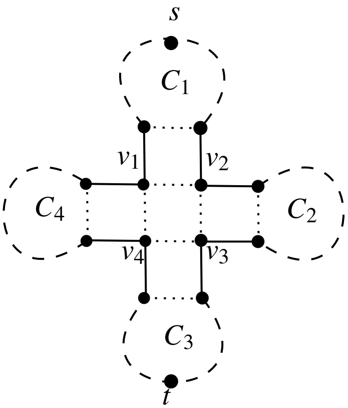
(F3)

ALP(G, s, t) satisfies (F3) if G contains four vertices v_1, v_2, v_3 and v_4, where each pair of them is a cutting pair, s lies on the boundary of G between v_1 and v_2, and t lies on the boundary of G between v_3 and v_4.

Forbidden Problems

(F3)

ALP(G, s, t) satisfies (F3) if G contains four vertices v_1, v_2, v_3 and v_4, where each pair of them is a cutting pair, s lies on the boundary of G between v_1 and v_2, and t lies on the boundary of G between v_3 and v_4.

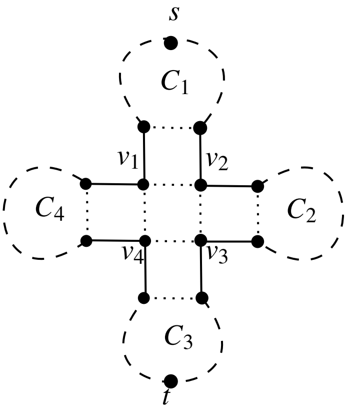


Forbidden Problems

(F3)

ALP(G, s, t) satisfies (F3) if G contains four vertices v_1, v_2, v_3 and v_4, where each pair of them is a cutting pair, s lies on the boundary of G between v_1 and v_2, and t lies on the boundary of G between v_3 and v_4.

- Any path between s and t cannot contain vertices from both C_2 and C_4 in the case of (F3).



Forbidden Problems

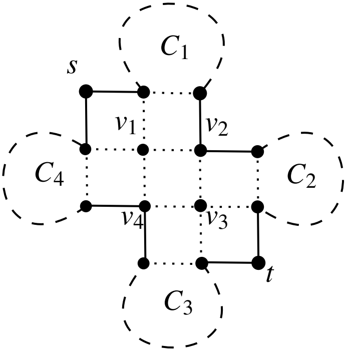
(F4)

ALP(G, s, t) satisfies (F4) if G contains four vertices v_1, v_2, v_3 and v_4 such that the structure of G around these vertices, s and t is as shown in the figure.

Forbidden Problems

(F4)

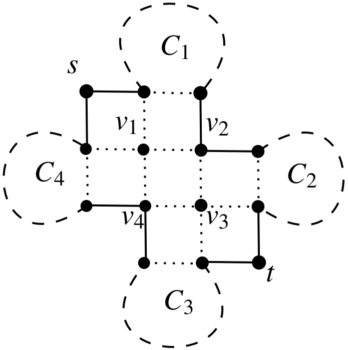
ALP(G, s, t) satisfies (F4) if G contains four vertices v_1, v_2, v_3 and v_4 such that the structure of G around these vertices, s and t is as shown in the figure.



Forbidden Problems

(F4)

ALP(G, s, t) satisfies (F4) if G contains four vertices v_1, v_2, v_3 and v_4 such that the structure of G around these vertices, s and t is as shown in the figure.



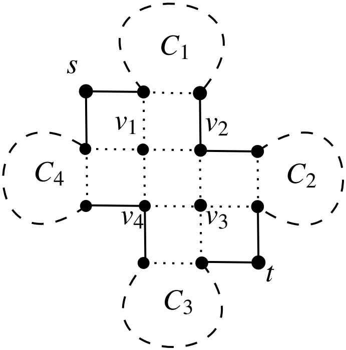
- Any path from s to t cannot contain vertices from all of C_1, C_2, C_3 and C_4 in the case of (F4).

Forbidden Problems

(F4)

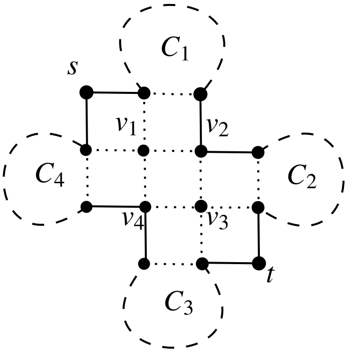
ALP(G, s, t) satisfies (F4) if G contains four vertices v_1, v_2, v_3 and v_4 such that the structure of G around these vertices, s and t is as shown in the figure.

- Any path from s to t cannot contain vertices from all of C_1, C_2, C_3 and C_4 in the case of (F4).
- There **is no Hamiltonian path** between s and t in G if ALP(G, s, t) satisfies any of conditions (C1), (C2), (F3) or (F4).



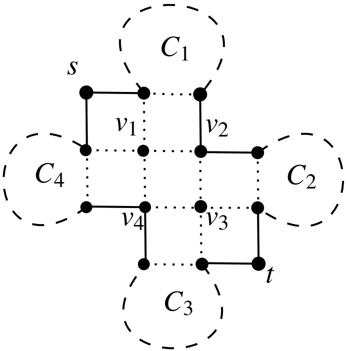
Solving the Forbidden Problems

- If problem $\mathrm{ALP}(G, s, t)$ satisfies (F4), the problem cannot satisfy any of the conditions (F1), (F2) or (F3) simultaneously, and the vertex set $\{v_1, v_2, v_3, v_4\}$ should be unique.



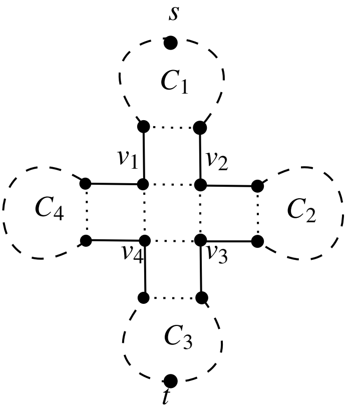
Solving the Forbidden Problems

- If problem ALP(G, s, t) satisfies (F4), the problem cannot satisfy any of the conditions (F1), (F2) or (F3) simultaneously, and the vertex set $\{v_1, v_2, v_3, v_4\}$ should be unique.
- We can convert it to a non-forbidden problem by removing the smallest subgraph among C_1, C_2, C_3 and C_4 from G.



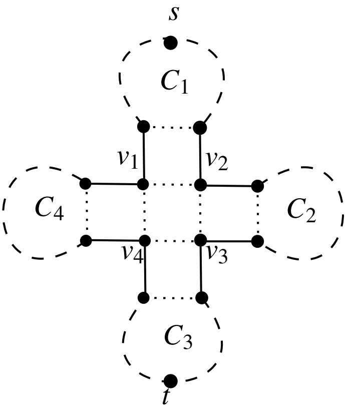
Solving the Forbidden Problems

- There may be more than one set of vertices $\{v_1, v_2, v_3, v_4\}$ which cause ALP(G, s, t) to satisfy (F3).



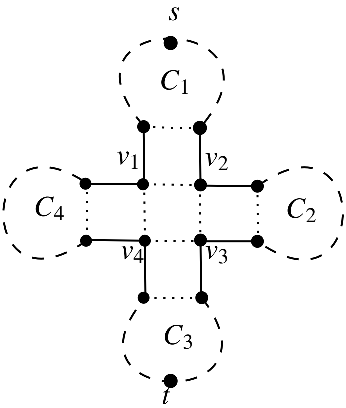
Solving the Forbidden Problems

- There may be more than one set of vertices $\{v_1, v_2, v_3, v_4\}$ which cause ALP(G, s, t) to satisfy (F3).
- Since such sets may not overlap, we can make the problem non-forbidden by removing the smallest subgraph among C_2 and C_4 from G for each such set of vertices.



Solving the Forbidden Problems

- There may be more than one set of vertices $\{v_1, v_2, v_3, v_4\}$ which cause ALP(G, s, t) to satisfy (F3).
- Since such sets may not overlap, we can make the problem non-forbidden by removing the smallest subgraph among C_2 and C_4 from G for each such set of vertices.
- ALP(G, s, t) can satisfy one or both of conditions (F1) and (F2) while satisfying (F3). In this case, we first convert the problem to the problem which only satisfies (F1) and/or (F2), and then convert it to a non-forbidden problem.



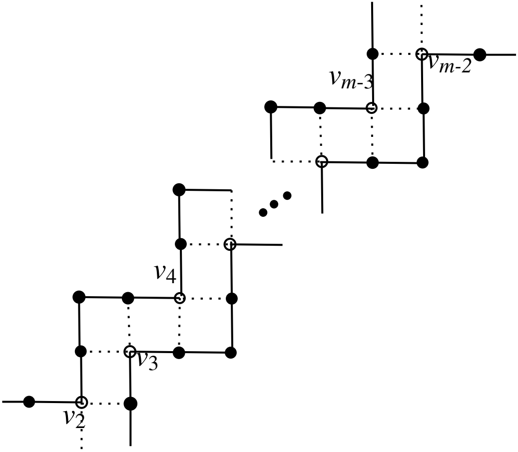
Solving the Forbidden Problems

- If ALP(G, s, t) satisfies only (F1) or (F2), similar to the previous cases, we can convert it to a non-forbidden problem by removing the smallest subgraph among the subgraphs $C_1, C_2, \ldots, C_{m+1}$ from G.

Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

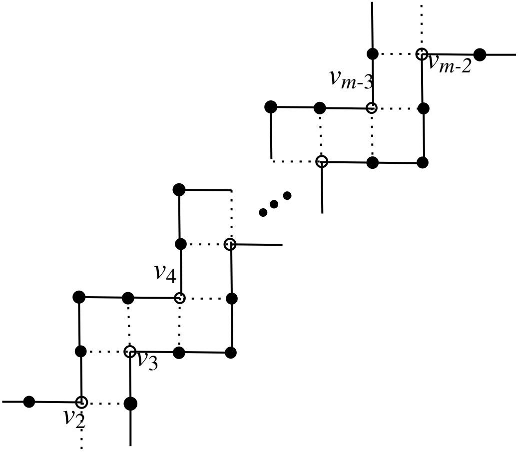


Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- Let (F1) or (F2) hold in ALP(G, s, t).

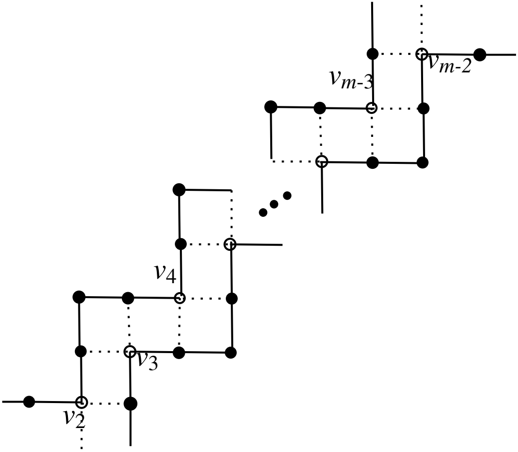


Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

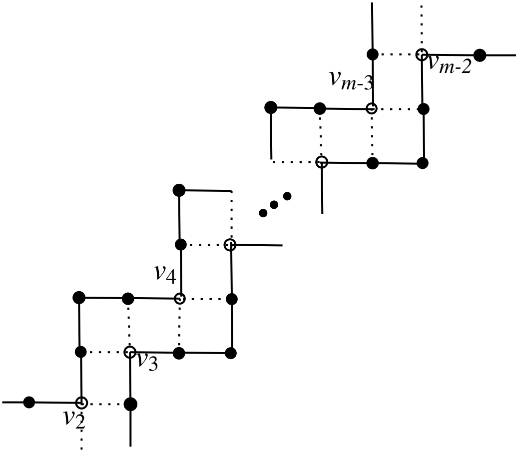
- Let (F1) or (F2) hold in ALP(G, s, t).
- If G contains two critical cutting pairs to s, then $m = 0$. Therefore, G must have only one critical cutting pair to s.



Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

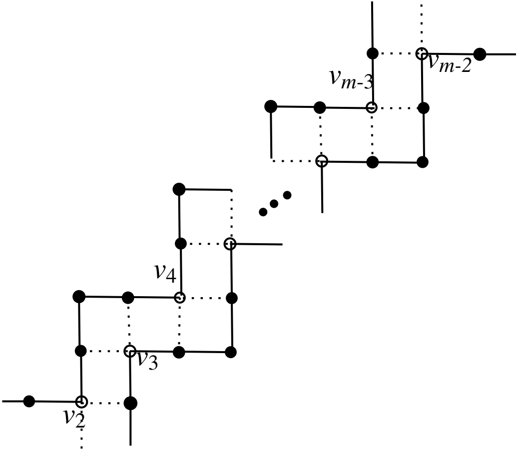


Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- Let v be the vertex of G cutting pair to s, distinct from t as otherwise $m = 1$.

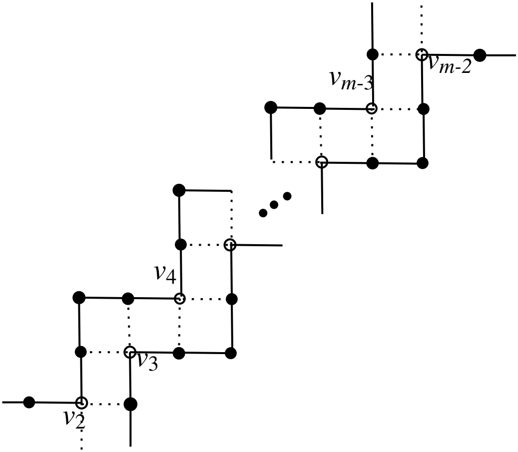


Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- Let v be the vertex of G cutting pair to s, distinct from t as otherwise $m = 1$.
- We use the following figures to analyse all possible cases.

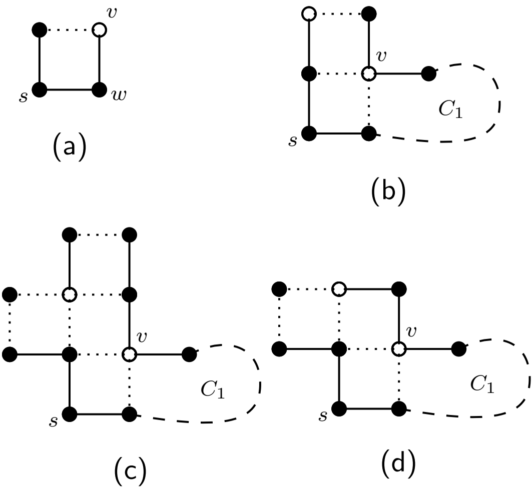


Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

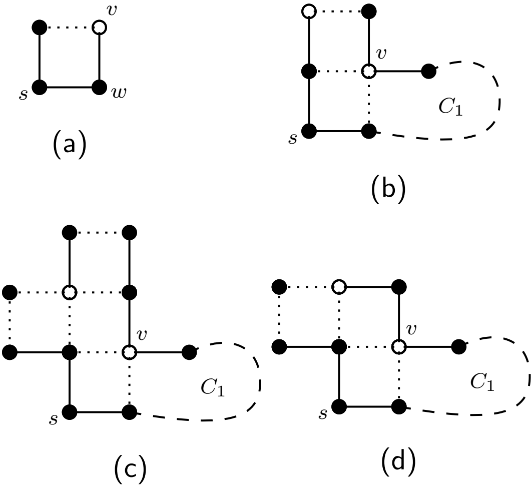
- When s is a degree-two boundary vertex, the possible configurations are depicted in figure.



Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.



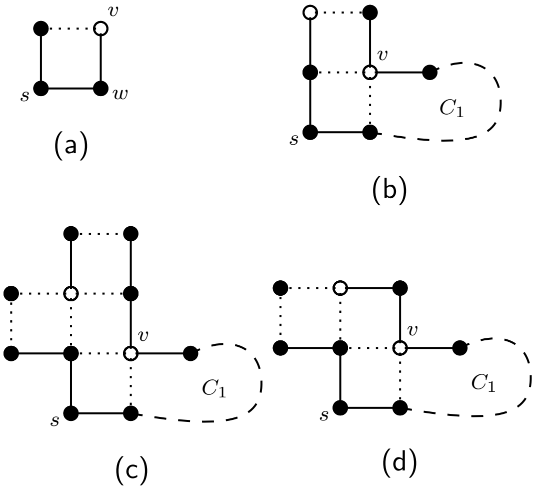
- When s is a degree-two boundary vertex, the possible configurations are depicted in figure.
- In Figure (a), one of the connected components of $G \setminus \{s, v\}$ has only one vertex, so (F1) or (F2) cannot hold.

Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

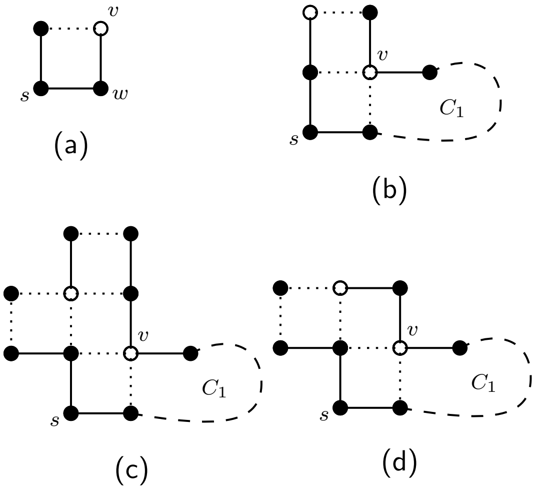
- For Figure (b) and (d), the structure of G_1 around v must be similar to the structure of G around s in Figure (a).



Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.



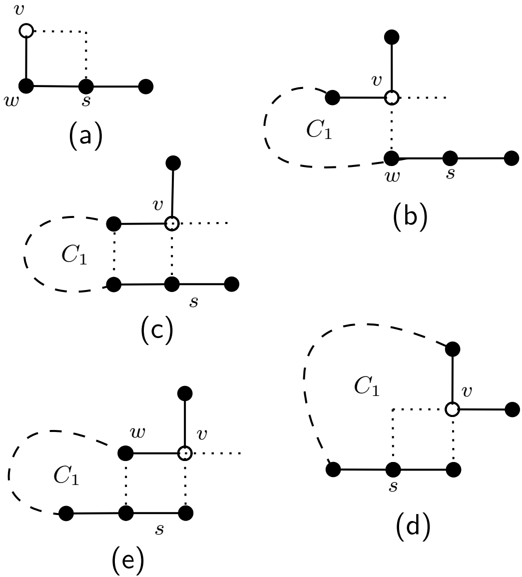
- For Figure (b) and (d), the structure of G_1 around v must be similar to the structure of G around s in Figure (a).
- For Figure (c), the structure of G_1 around v must be similar to the structure of G around s in Figure (b).

Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- When s is a degree-three boundary vertex, the possible configurations are depicted in figure.

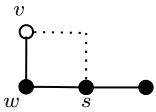


Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- When s is a degree-three boundary vertex, the possible configurations are depicted in figure.
- In Figure (a), (F1) or (F2) cannot hold because $G \setminus \{s, v\}$ has a connected component containing only one vertex.



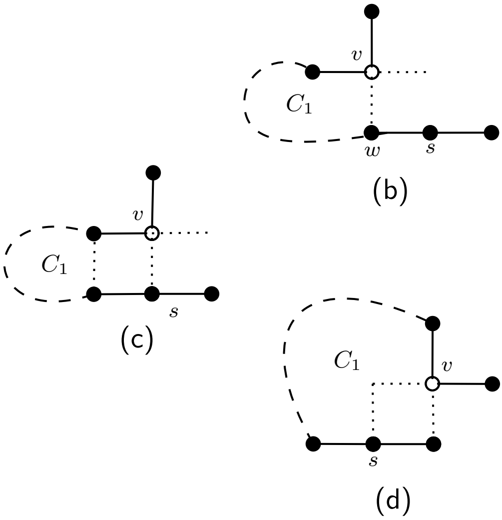
(a)

Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- In Figure (b)–(d), because v is a degree two boundary vertex of G_1, m is at most three.

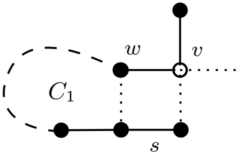


Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- In Figure (b)–(d), because v is a degree two boundary vertex of G_1, m is at most three.
- $m > 3$ is only possible in the case of Figure (e), which in this case the structure of G_1 around v must be similar to Figure (e) too.



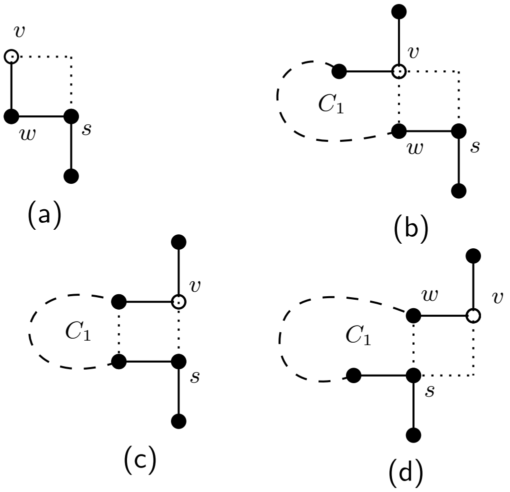
(e)

Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- When s is a degree-four boundary vertex, the possible configurations are depicted in figure.

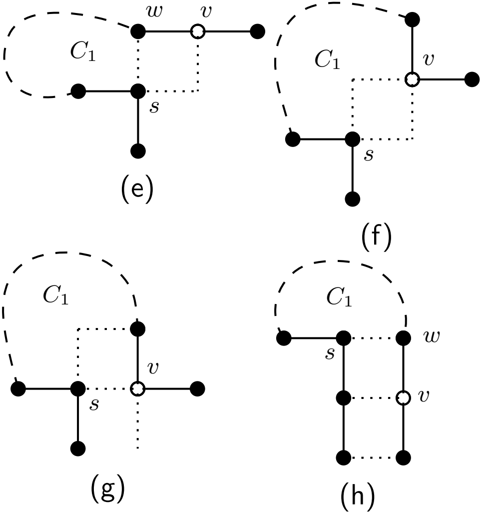


Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- When s is a degree-four boundary vertex, the possible configurations are depicted in figure.

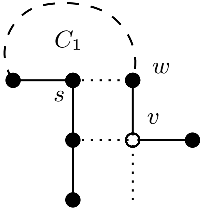


Solving the Forbidden Problems

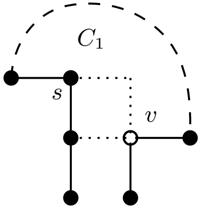
Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- When s is a degree-four boundary vertex, the possible configurations are depicted in figure.



(i)



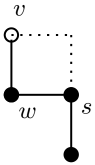
(j)

Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- When s is a degree-four boundary vertex, the possible configurations are depicted in figure.
- In Figure (a), (F1) or (F2) cannot hold because $G \setminus \{s, v\}$ has a connected component containing only one vertex.



(a)

Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

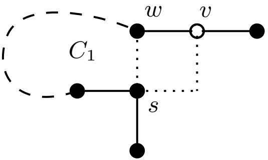
- In the other cases, vertex v must be a degree-two or a degree-three boundary vertex of G_1.

Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- In the other cases, vertex v must be a degree-two or a degree-three boundary vertex of G_1.
- Hence, either $m \leq 4$ or the structure of G_1 around v must be similar to Figure (e).



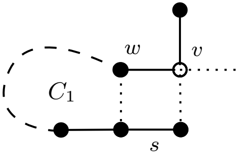
(e)

Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- $m > 4$ is possible only when the structure depicted in the figure is repeated in G_1, $\ldots$,G_{m-2}, respectively, around the vertices v_1, $\ldots$, v_{m-2}.



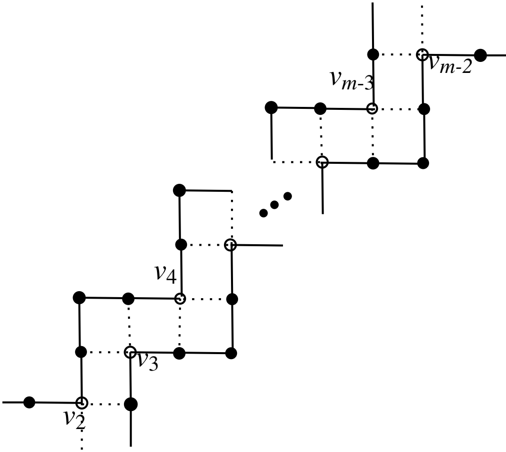
(e)

Solving the Forbidden Problems

Lemma

If ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$, then the structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ must be as illustrated in the figure shown.

- $m > 4$ is possible only when the structure depicted in the figure is repeated in G_1, $\ldots$,G_{m-2}, respectively, around the vertices v_1, $\ldots$, v_{m-2}.
- Therefore, when $m \geq 4$ the structure of G around the vertices $v_1, v_2, \ldots, v_{m-2}$ must be as depicted in the figure.

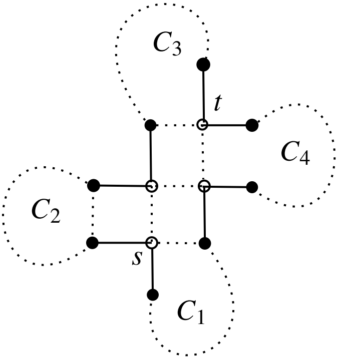


Solving the Forbidden Problems

- In some cases, both (F1) and (F2) can hold simultaneously.

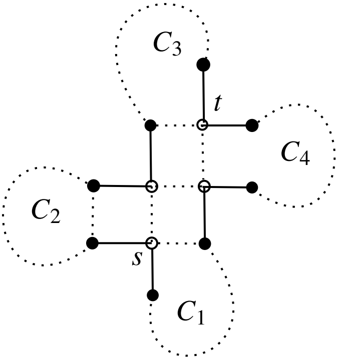
Solving the Forbidden Problems

- In some cases, both (F1) and (F2) can hold simultaneously.
- In this formation, removing one of C_1 or C_2, and one of C_3 and C_4 from G makes the problem non-forbidden.



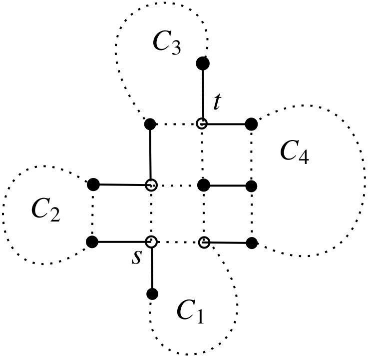
Solving the Forbidden Problems

- In some cases, both (F1) and (F2) can hold simultaneously.
- In this formation, removing one of C_1 or C_2, and one of C_3 and C_4 from G makes the problem non-forbidden.
- According to the lemmas, the smallest subgraph among C_1 and C_2, and the smallest subgraph among C_3 and C_4 should be removed



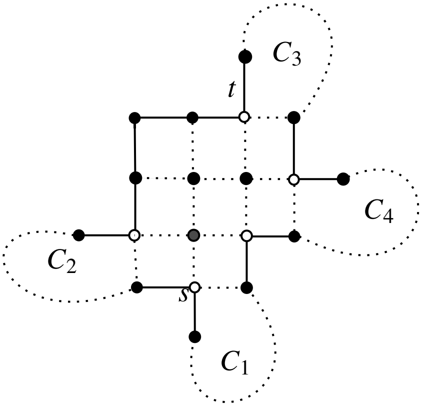
Solving the Forbidden Problems

- In this formation, removing the smallest subgraph among C_1 and C_3, and the smallest subgraph among C_1 and C_4, or C_2, from G makes the problem non-forbidden.



Solving the Forbidden Problems

- In this formation, removing the smallest subgraph among C_1 and C_3, and the smallest subgraph among C_1 and C_4, or C_2, from G makes the problem non-forbidden.
- In this formation, removing the smallest subgraph among C_2 and C_3, and the smallest subgraph among C_2 and C_4, or C_1, from G makes the problem non-forbidden.



Split Operations

- The split operations are used for dividing ALP(G, s, t) problem to smaller subproblems.

Split Operations

- The split operations are used for dividing ALP(G, s, t) problem to smaller subproblems.
- Using these operations, the problem is repeatedly split until $G \setminus B$ becomes 2-connected, where B is a boundary path of G between s and t.

Split Operations

- The split operations are used for dividing ALP(G, s, t) problem to smaller subproblems.
- Using these operations, the problem is repeatedly split until $G \setminus B$ becomes 2-connected, where B is a boundary path of G between s and t.
- In the split operations, ALP(G, s, t) is assumed not to be a forbidden problem, and is never split into forbidden subproblems.

Split Operations

- The split operations are used for dividing ALP(G, s, t) problem to smaller subproblems.
- Using these operations, the problem is repeatedly split until $G \setminus B$ becomes 2-connected, where B is a boundary path of G between s and t.
- In the split operations, ALP(G, s, t) is assumed not to be a forbidden problem, and is never split into forbidden subproblems.
- In each split operation, it is possible to construct a path between s and t containing at least $2/3$ of the nodes of G using the solutions of subproblems.

Type I Split Operation

- Type I split operation is done when G has a vertex cutting pair to s or t (without loss of generality say s).

Type I Split Operation

- Type I split operation is done when G has a vertex cutting pair to s or t (without loss of generality say s).
- When s and t are a cutting pair, since (F1) does not hold, a connected component of $G \setminus \{s, t\}$ has only one vertex w, and the problem can be solved by finding an approximated longest cycle containing the edge (s, w).

Type I Split Operation

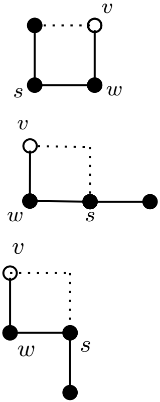
- Type I split operation is done when G has a vertex cutting pair to s or t (without loss of generality say s).
- When s and t are a cutting pair, since (F1) does not hold, a connected component of $G \setminus \{s, t\}$ has only one vertex w, and the problem can be solved by finding an approximated longest cycle containing the edge (s, w).
- Otherwise, first, let G contain only one vertex critical cutting pair to s, s and v be a cutting pair of G, and C_1 and C_2 be two connected components of $G \setminus \{s, v\}$ such that C_2 contains t.

Type I Split Operation

- Type I split operation is done when G has a vertex cutting pair to s or t (without loss of generality say s).
- When s and t are a cutting pair, since (F1) does not hold, a connected component of $G \setminus \{s,t\}$ has only one vertex w, and the problem can be solved by finding an approximated longest cycle containing the edge (s,w).
- Otherwise, first, let G contain only one vertex critical cutting pair to s, s and v be a cutting pair of G, and C_1 and C_2 be two connected components of $G \setminus \{s,v\}$ such that C_2 contains t.
- We illustrated all the possible cases for G around s and v, when s is a degree-two, degree-three and degree-four boundary vertex.

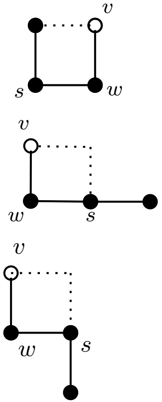
Type I Split Operation

- **All The Cases Except The Cases of Figures**: If $C_1 \cup \{s, v\}$ is not 2-connected, let w be the common adjacent vertex of s and v in C_1.



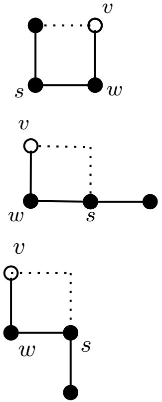
Type I Split Operation

- **All The Cases Except The Cases of Figures**: If $C_1 \cup \{s, v\}$ is not 2-connected, let w be the common adjacent vertex of s and v in C_1.
- ALP(G, s, t) can be split into ALP$(C_2 \cup \{v\}, v, t)$ and one of the three subproblems ALP$(C_1 \cup \{s, v\}, s, v)$, ALP$(C_1 \cup \{s\}, s, w)$ or ALP$(C_1 \cup \{v\}, v, w)$.



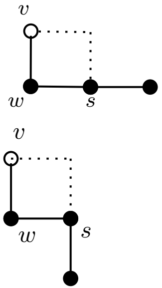
Type I Split Operation

- **All The Cases Except The Cases of Figures**: If $C_1 \cup \{s, v\}$ is not 2-connected, let w be the common adjacent vertex of s and v in C_1.
- ALP(G, s, t) can be split into ALP$(C_2 \cup \{v\}, v, t)$ and one of the three subproblems ALP$(C_1 \cup \{s, v\}, s, v)$, ALP$(C_1 \cup \{s\}, s, w)$ or ALP$(C_1 \cup \{v\}, v, w)$.
- Concatenate the solutions of subproblems and one of the edges (w, v) or (s, v) if needed.



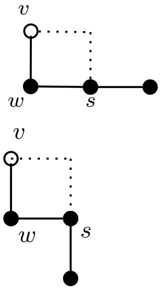
Type I Split Operation

- **The Cases of Figures**: C_1 contains only one vertex w.



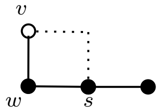
Type I Split Operation

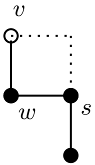
- **The Cases of Figures**: C_1 contains only one vertex w.
- If (C1) or (C2) do not hold, concatenating the edges (s, w) and (w, v) and the solution of ALP$(C_2 \cup \{v\}, v, t)$ will result in a solution for ALP(G, s, t).



Type I Split Operation

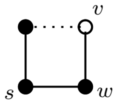
- **The Cases of Figures**: C_1 contains only one vertex w.
- If (C1) or (C2) do not hold, concatenating the edges (s, w) and (w, v) and the solution of ALP$(C_2 \cup \{v\}, v, t)$ will result in a solution for ALP(G, s, t).
- Otherwise, ALP$(G \smallsetminus C_1, s, t)$ is solved instead.





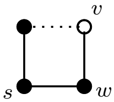
Type I Split Operation

- **The Cases of Figures**: C_1 contains only one vertex w.



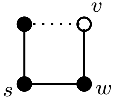
Type I Split Operation

- **The Cases of Figures**: C_1 contains only one vertex w.
- If (C1) or (C2) do not hold, concatenating the edges (s, w) and (w, v) and the solution of ALP$(C_2 \cup \{v\}, v, t)$ will result in a solution for ALP(G, s, t).



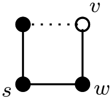
Type I Split Operation

- **The Cases of Figures**: C_1 contains only one vertex w.
- If (C1) or (C2) do not hold, concatenating the edges (s, w) and (w, v) and the solution of ALP$(C_2 \cup \{v\}, v, t)$ will result in a solution for ALP(G, s, t).
- If (C1) or (C2) hold, the structure of G around t is assumed to be similar to its structure around s.



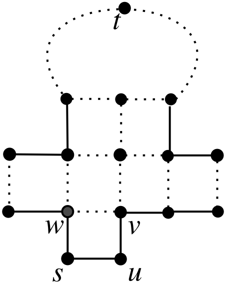
Type I Split Operation

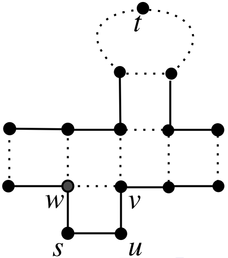
- **The Cases of Figures**: C_1 contains only one vertex w.
- If (C1) or (C2) do not hold, concatenating the edges (s, w) and (w, v) and the solution of ALP$(C_2 \cup \{v\}, v, t)$ will result in a solution for ALP(G, s, t).
- If (C1) or (C2) hold, the structure of G around t is assumed to be similar to its structure around s.
- Therefore, ALP$(C_2 \cup \{v\}, v, t)$ cannot satisfy (F1).



Type I Split Operation

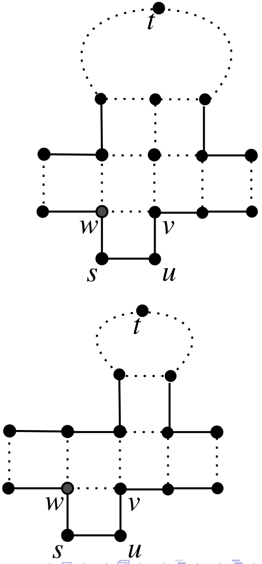
- Because (F3) does not hold in ALP(G, s, t), ALP$(C_2 \cup \{v\}, v, t)$ can satisfy (F2) only if the structure of G around s is as illustrated in the Figures.





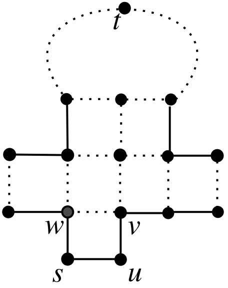
Type I Split Operation

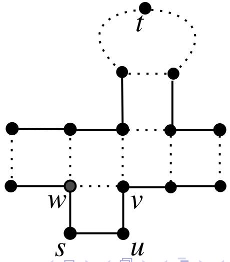
- Because (F3) does not hold in ALP(G, s, t), ALP$(C_2 \cup \{v\}, v, t)$ can satisfy (F2) only if the structure of G around s is as illustrated in the Figures.
- In these two cases, ALP$(C_2 \cup \{v\}, w, t)$ does not satisfy (F1) and (F2).



Type I Split Operation

- Because (F3) does not hold in ALP(G, s, t), ALP$(C_2 \cup \{v\}, v, t)$ can satisfy (F2) only if the structure of G around s is as illustrated in the Figures.
- In these two cases, ALP$(C_2 \cup \{v\}, w, t)$ does not satisfy (F1) and (F2).
- Therefore, the problem can be solved by solving ALP$(C_2 \cup \{v\}, v, t)$ or ALP$(C_2 \cup \{v\}, w, t)$ and adding the edges (s, u) and (u, v), or (s, w), respectively.





Type I Split Operation

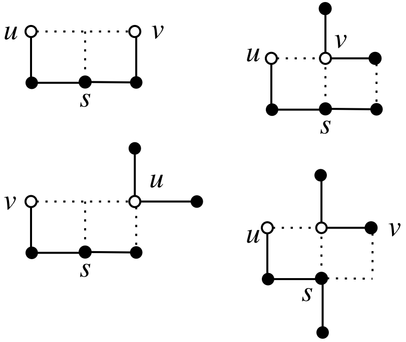
- Assume that G contains two vertices u and v critical cutting pairs to s, and C_u and C_v are the connected components of, respectively, $G \setminus \{s, u\}$ and $G \setminus \{s, v\}$ which does not contain t.

Type I Split Operation

- Assume that G contains two vertices u and v critical cutting pairs to s, and C_u and C_v are the connected components of, respectively, $G \setminus \{s, u\}$ and $G \setminus \{s, v\}$ which does not contain t.
- In this case, C_u or C_v (without loss of generality say C_u) should contains only one vertex w because (F2) does not hold.

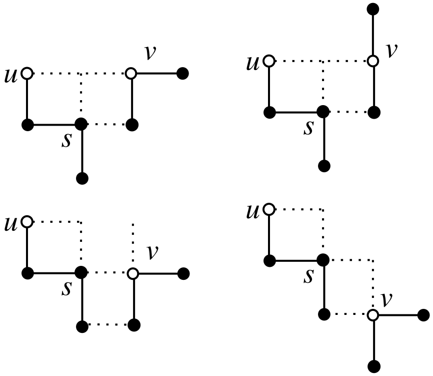
Type I Split Operation

- Assume that G contains two vertices u and v critical cutting pairs to s, and C_u and C_v are the connected components of, respectively, $G \setminus \{s, u\}$ and $G \setminus \{s, v\}$ which does not contain t.
- In this case, C_u or C_v (without loss of generality say C_u) should contains only one vertex w because (F2) does not hold.
- All the possible cases for G around s are shown in Figure



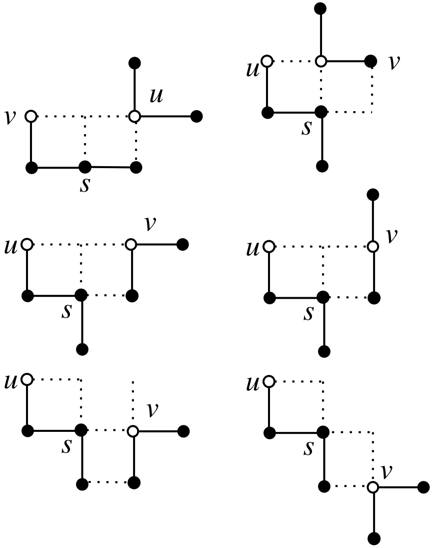
Type I Split Operation

- Assume that G contains two vertices u and v critical cutting pairs to s, and C_u and C_v are the connected components of, respectively, $G \setminus \{s, u\}$ and $G \setminus \{s, v\}$ which does not contain t.
- In this case, C_u or C_v (without loss of generality say C_u) should contains only one vertex w because (F2) does not hold.
- All the possible cases for G around s are shown in Figure



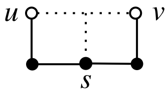
Type I Split Operation

- **The cases shown in the figures:** By removing the only vertex of C_u from G the problem can be converted to the case that s has only one critical cutting pair.



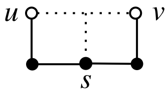
Type I Split Operation

- **The case shown in the figure:** ALP$(G \setminus C_u, s, t)$ may satisfy (F3).



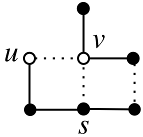
Type I Split Operation

- **The case shown in the figure:** ALP$(G \setminus C_u, s, t)$ may satisfy (F3).
- One of $ALP(G \setminus C_u, s, t)$ or ALP$(G \setminus C_v, s, t)$ can be solved which does not satisfy (F3) instead of ALP(G, s, t).



Type I Split Operation

- **The case shown in the figure:** The problem can be split into $\text{ALP}(C_v \cup \{s, v\}, s, v)$ and $\text{ALP}(G \setminus (C_v \cup C_u \cup \{s\}), v, t)$.



Type II Split Operation

- Type II split is used to split ALP(G, s, t) when type I split is not possible and there is a cutting pair u and v in G whose removal disconnect s and t.

Type II Split Operation

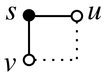
- Type II split is used to split ALP(G, s, t) when type I split is not possible and there is a cutting pair u and v in G whose removal disconnect s and t.
- u and v should lie on different boundary paths B_1 and B_2 between s and t.

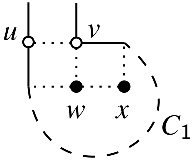
Type II Split Operation

- Type II split is used to split ALP(G, s, t) when type I split is not possible and there is a cutting pair u and v in G whose removal disconnect s and t.
- u and v should lie on different boundary paths B_1 and B_2 between s and t.
- Let u be the nearest vertex to s along B_1 which is a cutting pair to a vertex of B_2, and v be the cutting pair to u nearest to s along B_2.

Type II Split Operation

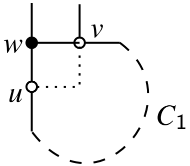
- Type II split is used to split ALP(G, s, t) when type I split is not possible and there is a cutting pair u and v in G whose removal disconnect s and t.
- u and v should lie on different boundary paths B_1 and B_2 between s and t.
- Let u be the nearest vertex to s along B_1 which is a cutting pair to a vertex of B_2, and v be the cutting pair to u nearest to s along B_2.
- Let C_1 and C_2 be the connected components of $G \setminus \{u, v\}$ which, respectively, contains s and t.

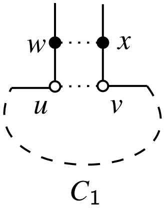




Type II Split Operation

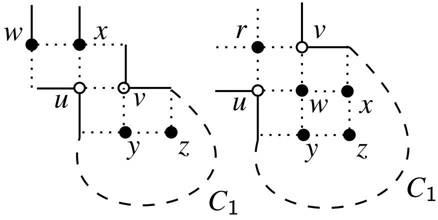
- Type II split is used to split ALP(G, s, t) when type I split is not possible and there is a cutting pair u and v in G whose removal disconnect s and t.
- u and v should lie on different boundary paths B_1 and B_2 between s and t.
- Let u be the nearest vertex to s along B_1 which is a cutting pair to a vertex of B_2, and v be the cutting pair to u nearest to s along B_2.
- Let C_1 and C_2 be the connected components of $G \setminus \{u, v\}$ which, respectively, contains s and t.



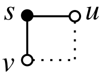


Type II Split Operation

- Type II split is used to split ALP(G, s, t) when type I split is not possible and there is a cutting pair u and v in G whose removal disconnect s and t.
- u and v should lie on different boundary paths B_1 and B_2 between s and t.
- Let u be the nearest vertex to s along B_1 which is a cutting pair to a vertex of B_2, and v be the cutting pair to u nearest to s along B_2.
- Let C_1 and C_2 be the connected components of $G \setminus \{u, v\}$ which, respectively, contains s and t.



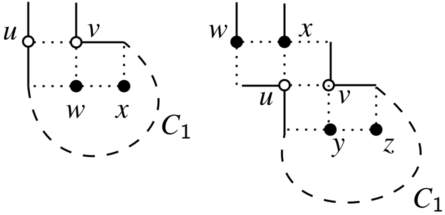
Type II Split Operation



- **The Case Shown in the Figure**: The existence of the cutting pair u and v will not cause $G \setminus B1$ or $G \setminus B2$ to be not 2-connected, so the problem need not be split.

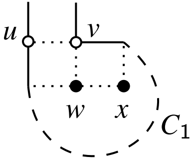
Type II Split Operation

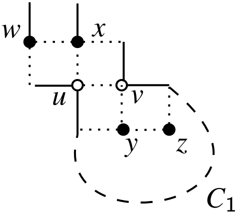
- **The Cases Shown in the Figures**: The problem is split into ALP$(C_1 \cup \{u, v\}, s, v)$ and ALP$(C_2 \cup \{u, v\}, u, t)$. Because u and v in the subproblems are type I boundary vertices, (F1) and (F2) cannot hold in the subproblems.



Type II Split Operation

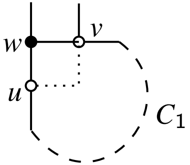
- **The Cases Shown in the Figures**: The problem is split into ALP($C_1 \cup \{u, v\}, s, v$) and ALP($C_2 \cup \{u, v\}, u, t$). Because u and v in the subproblems are type I boundary vertices, (F1) and (F2) cannot hold in the subproblems.
- Then the paths from two subproblems are merged.

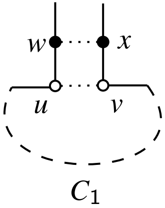




Type II Split Operation

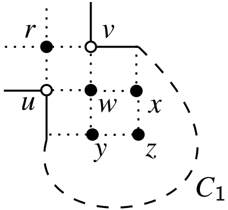
- **The Cases Shown in the Figures**: The problem is split into ALP$(C_1 \cup \{u, v\}, s, v)$ and ALP$(C_2 \cup \{u, v\}, u, t)$. Because u and v in the subproblems are type I boundary vertices, (F1) and (F2) cannot hold in the subproblems.
- Then the paths from two subproblems are merged.
- The split operations for these cases are done in the similar way.





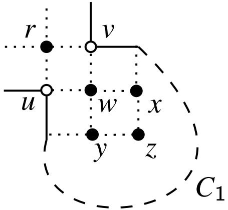
Type II Split Operation

- **The Case Shown in the Figure**: w cannot be a boundary vertex because of the definition of u and v.



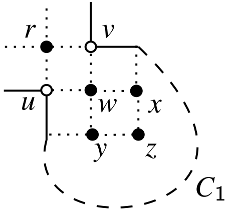
Type II Split Operation

- **The Case Shown in the Figure**: w cannot be a boundary vertex because of the definition of u and v.
- It can be assumed that r is not a boundary vertex, as otherwise only ALP(G, t, s) is needed to be solved instead of ALP(G, s, t).



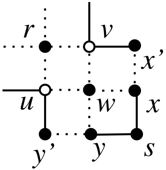
Type II Split Operation

- **The Case Shown in the Figure**: w cannot be a boundary vertex because of the definition of u and v.
- It can be assumed that r is not a boundary vertex, as otherwise only ALP(G, t, s) is needed to be solved instead of ALP(G, s, t).
- If at least one of x or y, is a non-boundary vertex, both of $C_1 \cup \{u\}$ and $C_2 \cup \{u, v\}$ must be 2-connected, the problem can be split into ALP$(C_1 \cup \{u\}, s, u)$ and ALP$(C_2 \cup \{u, v\}, u, t)$.



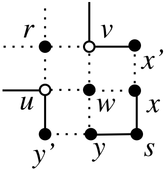
Type II Split Operation

- When both of x and y are boundary vertices, the only possible case is illustrated in the figure.



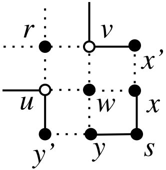
Type II Split Operation

- When both of x and y are boundary vertices, the only possible case is illustrated in the figure.
- If both edges (x, x') and (y, y') are boundary edges, since (F4) does not hold, ALP(G, t, s) can be solved instead.



Type II Split Operation

- When both of x and y are boundary vertices, the only possible case is illustrated in the figure.
- If both edges (x, x') and (y, y') are boundary edges, since (F4) does not hold, ALP(G, t, s) can be solved instead.
- It can be assumed that (x, x') is a non-boundary edge, and the problem can be solved by solving ALP$(C_2 \cup \{u, v\}, u, t)$ and finding a cycle in the 2-connected graph $C_1 \cup \{u\} \setminus \{s, x, x'\}$ containing the edge (u, w) and then removing (u, w) and adding the edges (s, x) and (x, w).



Type III Split Operations

- When type I and type II split operations are not possible and $G \setminus B$ is not 2-connected, in which B is a boundary path between s and t, type III split operation is used to split the problem.

Type III Split Operations

- When type I and type II split operations are not possible and $G \setminus B$ is not 2-connected, in which B is a boundary path between s and t, type III split operation is used to split the problem.
- Considering the preconditions of type I and type II split operations, either B contains a cutting pair of G or G has a cutting set of three vertices two of which are in B.

Type III Split Operations

- When type I and type II split operations are not possible and $G \setminus B$ is not 2-connected, in which B is a boundary path between s and t, type III split operation is used to split the problem.
- Considering the preconditions of type I and type II split operations, either B contains a cutting pair of G or G has a cutting set of three vertices two of which are in B.
- Let v be the nearest vertex to s along the path B that is in such a cutting set of vertices. Clearly, v is different from s, since type I split is not possible.

Type III Split Operations

- Let there be a vertex u such that u and v are a cutting pair of G.

Type III Split Operations

- Let there be a vertex u such that u and v are a cutting pair of G.
- It can be shown that ALP(G', v, t) is not a forbidden problem, in which G' is the result of removing the vertices of B_{sv} except v from G;

Type III Split Operations

- Let there be a vertex u such that u and v are a cutting pair of G.
- It can be shown that ALP(G', v, t) is not a forbidden problem, in which G' is the result of removing the vertices of B_{sv} except v from G;
- The problem can be solved by concatenating B_{sv} and the solution of ALP(G', v, t).

Type III Split Operations

- Knowing that B_{vt} is a boundary path between v and t in G', let B' be the other boundary path of G' between v and t.

Type III Split Operations

- Knowing that B_{vt} is a boundary path between v and t in G', let B' be the other boundary path of G' between v and t.
- No vertex of B_{vt} could be a cutting pair to a vertex of B', as otherwise either type I or type II split operations are possible on ALP(G, s, t),or,

Type III Split Operations

- Knowing that B_{vt} is a boundary path between v and t in G', let B' be the other boundary path of G' between v and t.
- No vertex of B_{vt} could be a cutting pair to a vertex of B', as otherwise either type I or type II split operations are possible on ALP(G, s, t),or,
- B_{sv} contains a vertex different from v which is in a cutting set of three vertices two of which are in B.

Type III Split Operations

- Knowing that B_{vt} is a boundary path between v and t in G', let B' be the other boundary path of G' between v and t.
- No vertex of B_{vt} could be a cutting pair to a vertex of B', as otherwise either type I or type II split operations are possible on ALP(G, s, t),or,
- B_{sv} contains a vertex different from v which is in a cutting set of three vertices two of which are in B.
- Both cases leading contradictions. Therefore, none of the conditions (C1), (C2), (F3) or (F4) hold on ALP(G', v, t).

Type III Split Operations

- Let v be a vertex of a cutting set of three vertices two of which are in B, and u be the vertex of B next to v.

Type III Split Operations

- Let v be a vertex of a cutting set of three vertices two of which are in B, and u be the vertex of B next to v.
- Considering the preconditions of this split operation and the previous case, either $G \setminus B_{sv}$ or $G \setminus B_{su}$ should be 2-connected.

Type III Split Operations

- The problem can be solved by concatenating the solution of ALP$(G \setminus B_{sv,} u, t)$ and B_{su} when $G \setminus B_{sv}$ is 2-connected, and the solution of ALP$(G \setminus B_{su}, u', t)$ and $B_{su'}$ when $G \setminus B_{su}$ is 2-connected, in which u' is the vertex of B next to u.

Type III Split Operations

- The problem can be solved by concatenating the solution of ALP$(G \setminus B_{sv,} u, t)$ and B_{su} when $G \setminus B_{sv}$ is 2-connected, and the solution of ALP$(G \setminus B_{su}, u', t)$ and $B_{su'}$ when $G \setminus B_{su}$ is 2-connected, in which u' is the vertex of B next to u.

Type III Split Operations

- The problem can be solved by concatenating the solution of ALP$(G \smallsetminus B_{sv,} u, t)$ and B_{su} when $G \smallsetminus B_{sv}$ is 2-connected, and the solution of ALP$(G \smallsetminus B_{su}, u', t)$ and $B_{su'}$ when $G \smallsetminus B_{su}$ is 2-connected, in which u' is the vertex of B next to u.
- The argumentation that the subproblem ALP$(G \smallsetminus B_{sv}, u, t)$ or ALP$(G \smallsetminus B_{su}, u', t)$ is not forbidden is similar to the previous case.

Algorithm ALP(G,s,t)

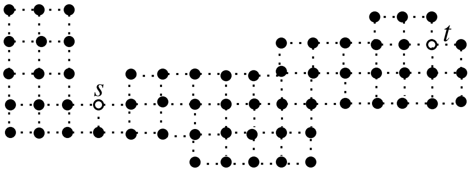
Algorithm 1 The algorithm for solving $ALP(G,\ s,\ t)$

Input: A 2-connected solid grid graph G and its boundary vertices s and t

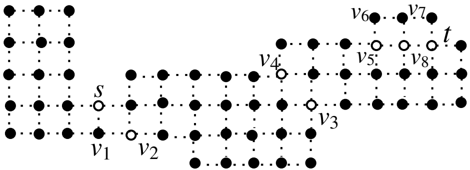
Output: A path between s and t of length at least $2/3$ of the length of the longest path between s and t

if *ALP(G,s,t) is forbidden* **then**
  solve the problem as explained before
else if *s and t are incident to a common boundary edge of G* **then**
  solve the problem using the longest cycle approximation algorithm
else if *one of type I, type II, or type III splits are possible* **then**
  split the problem and solve it recursively
else
  let B be the boundary path of G between s and t
  using a pair of parallel edges, merge B by a long cycle of $G \setminus B$

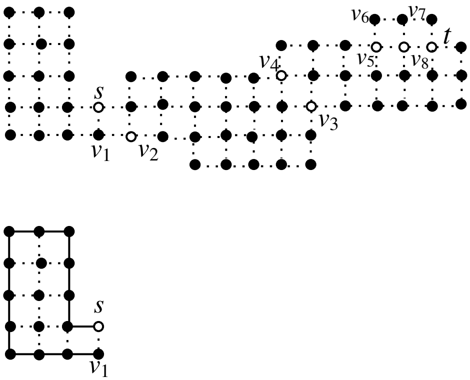
- This is the problem, a solid grid graph and two vertices s and t. We want to find the longest path between s and t using our algorithm.

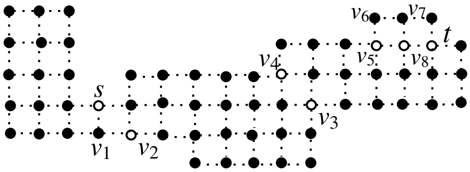


- We see the solid grid graph along with the cut vertices.

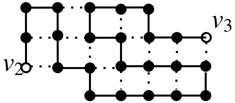


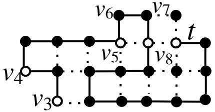
- First, the algorithm splits the problem using vertex s and its critical cutting pair v_2 (type I split).One of the resulting subproblems is shown here.



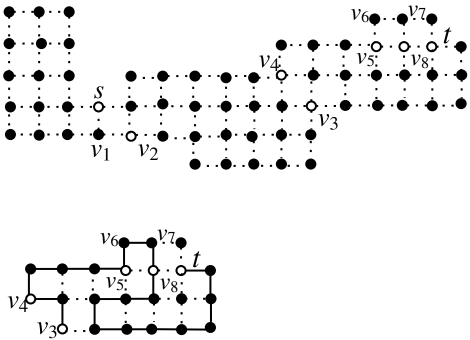


- The other one is split by the algorithm using the vertices v_3 and v_4 (type II split) into two subproblems illustrated by the bottom two graphs.



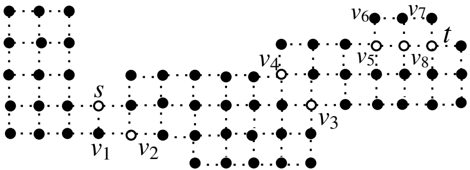


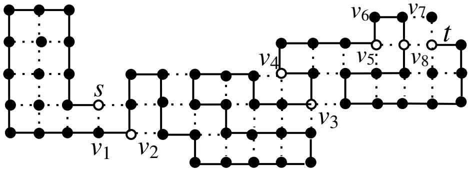
- Finally, using the cutting set $\{v_5,v_8,t\}$, the algorithm splits this subproblem (type III split).



...

- This illustrates the resulting long path between s and t, obtained by concatenating the solutions of subproblems.





Lemma:There is a polynomial-time $2/3$ -approximation algorithm for the version of the longest path problem in solid grid graphs in which the end-vertices are specified and forced to be on the boundary.

Lemma: There is a polynomial-time $2/3$ -approximation algorithm for the version of the longest path problem in solid grid graphs in which the end-vertices are specified and forced to be on the boundary.

Let G be a solid grid graph. If G is not 2-connected, it should contain a cut vertex v. A path between two given vertices s and t in G cannot pass through the vertices of the connected components of $G\backslash v$ not containing s and t. Moreover, any path between s and t must pass through v, if s and t are in different connected components of $G\backslash v$. Therefore, when G is not 2-connected, one can split the problem easily using the cut vertices of G. Thus, without loss of generality, we assume that G is 2-connected, and reduce the problem to $ALP(G, s, t)$.

Lemma:There is a polynomial-time $2/3$ -approximation algorithm for the version of the longest path problem in solid grid graphs in which the end-vertices are specified and forced to be on the boundary.

Previously we showed how we can convert a forbidden problem to a non-forbidden one by removing some of its vertices.We also showed that removing these vertices does not reduce the approximation ratio of our algorithm in the case of forbidden problems.we showed that we never create a forbidden subproblem while spliting.So during recursions we never encounter forbidden problem. In the case when G contains a boundary edge (s,t), we can find a cycle of G containing (s, t) and remove the edge (s, t) to create a path between s and t of length at least $2|G|/3$.

Lemma:There is a polynomial-time $2/3$ -approximation algorithm for the version of the longest path problem in solid grid graphs in which the end-vertices are specified and forced to be on the boundary.

Previously we showed how we can convert a forbidden problem to a non-forbidden one by removing some of its vertices.We also showed that removing these vertices does not reduce the approximation ratio of our algorithm in the case of forbidden problems.we showed that we never create a forbidden subproblem while spliting.So during recursions we never encounter forbidden problem. In the case when G contains a boundary edge (s,t), we can find a cycle of G containing (s, t) and remove the edge (s, t) to create a path between s and t of length at least $2|G|/3$.

If one of the split operations is possible, we split and solve recursively.We showed before split operations create consistent and non-forbidden problems. Finally, if none of the above cases hold on $ALP(G, s, t)$, $G\backslash B$ must be 2-connected and there must be a pair of parallel edges $e_1 \epsilon B$ and $e_2 \epsilon G\backslash B$. In this case,we find a long cycle in $G\backslash B$ containing the edge e_2 and merge it with B. So the algorithm works correctly.

Lemma: There is a polynomial-time $2/3$ -approximation algorithm for the version of the longest path problem in solid grid graphs in which the end-vertices are specified and forced to be on the boundary.

The longest cycle approximation algorithm in this case runs in linear time. Also, one can implement all the steps of our algorithm, except the recursive calls, to run in linear time with respect to n, the size of graph G. Thus, the worst case time complexity of Algorithm $ALP(G,s,t)$ is $O(n^2)$, because the total number of vertices of the subproblems created in split operations is always less than the number of the vertices of the graph G.

Approximation Ratio

Lemma:There is a polynomial-time $2/3$ -approximation algorithm for the longest path problem in solid grid graphs

Approximation Ratio

Lemma:There is a polynomial-time $2/3$ -approximation algorithm for the longest path problem in solid grid graphs

Our approximation algorithm for finding a longest path P' is as follows: for any pair of boundary vertices u and v of G find a long path using the algorithm of Lemma 5.1, and let P' be the path of maximum length among these paths. Because there are at most n^2 such pairs, the path P' can be found in polynomial time and we need to prove that $|P'|$ is atleast $2|P|/3$.

Approximation Ratio

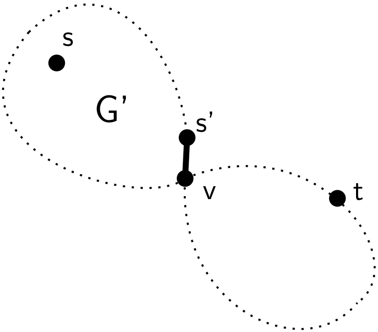
Lemma:There is a polynomial-time $2/3$ -approximation algorithm for the longest path problem in solid grid graphs

Our approximation algorithm for finding a longest path P' is as follows: for any pair of boundary vertices u and v of G find a long path using the algorithm of Lemma 5.1, and let P' be the path of maximum length among these paths. Because there are at most n^2 such pairs, the path P' can be found in polynomial time and we need to prove that $|P'|$ is atleast $2|P|/3$.

If both s and t are boundary vertices, we have $|P'| > 2|P|/3$. If both s and t are in the same 2-connected maximal subgraph G' of G, then G' should contain all vertices of P and atleast two boundary vertices u and v. Then the longest path found by the algorithm must be at least $2|G'|/3$.Then, we have $|P'| \geq 2|G'|/3 \geq 2|P|/3$.

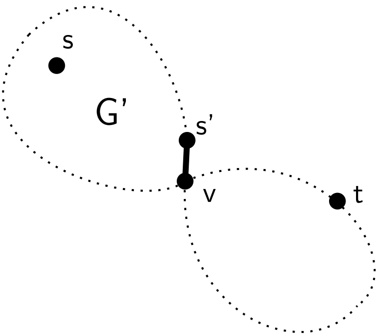
Approximation ratio Case 1

- Only one of s and t, in this case t is a boundary vertex.



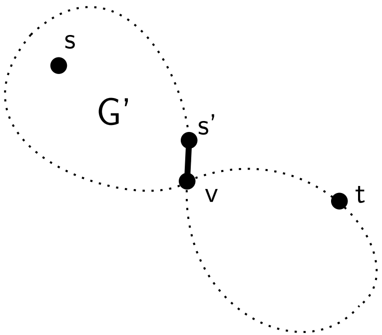
Approximation ratio Case 1

- Only one of s and t, in this case t is a boundary vertex.
- let v be the nearest cut vertex of G to s along path P, G' be the connected component of $G \setminus v$ containing s, and s' be a boundary vertex of G adjacent to v in G'.



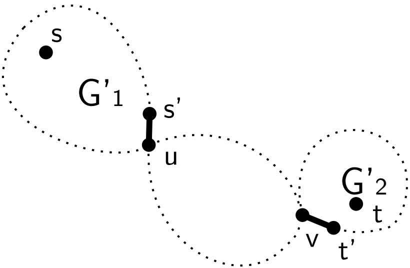
Approximation ratio Case 1

- Only one of s and t, in this case t is a boundary vertex.
- let v be the nearest cut vertex of G to s along path P, G' be the connected component of $G \setminus v$ containing s, and s' be a boundary vertex of G adjacent to v in G'.
- Because s', v and t are boundary vertices, and s' and v are adjacent, we can argue that the length of the long path found between s' and t by the algorithm is at least $2|P|/3$.



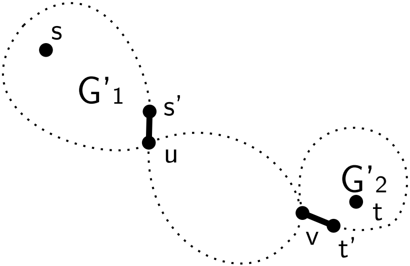
Approximation ratio Case 2

- Both s and t are non boundary vertices.



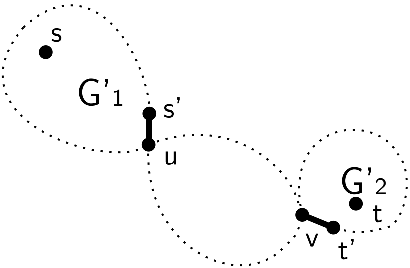
Approximation ratio Case 2

- Both s and t are non boundary vertices.
- let u and v be the nearest cut vertex of G to s and t along path P, G'_1 and G'_2 be the connected components of $G \backslash u, v$ containing s and t, and s' and t' be the boundary vertices of G adjacent to u and v in G'_1 and G'_2.



Approximation ratio Case 2

- Both s and t are non boundary vertices.
- let u and v be the nearest cut vertex of G to s and t along path P, G'_1 and G'_2 be the connected components of $G\backslash u, v$ containing s and t, and s' and t' be the boundary vertices of G adjacent to u and v in G'_1 and G'_2.
- Because s',u,v and t' are boundary vertices, and s' and u are adjacent and v and t' are adjacent,the length of the long path found between s' and t' is at least $2|P|/3$.



References I

Ryuhei Uehara and Yushi Uno.
Efficient algorithms for the longest path problem.
In Rudolf Fleischer and Gerhard Trippen, editors, *Algorithms and Computation*, pages 871–883, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy.
Proof verification and the hardness of approximation problems.
J. ACM, 45(3):501–555, May 1998.

D. Karger, R. Motwani, and G. D. S. Ramkumar.
On approximating the longest path in a graph.
Algorithmica, 18(1):82–98, May 1997.

References II

Asghar Asgharian Sardroud and Alireza Bagheri.
An approximation algorithm for the longest path problem in solid grid graphs.
Optimization Methods and Software, 31(3):479–493, 2016.