On Strings Having the Same Length-k Substrings

CPM 2022

Giulia Bernardini[1,5], Alessio Conte[2], **Estéban Gabory**[1], Roberto Grossi[2], Grigorios Loukides[3], Solon P. Pissis[1,4], Giulia Punzi[2] and Michelle Sweering[1]

[1]CWI Amsterdam, [2]Università di Pisa, [3]King's College London, [4]Vrije Universiteit Amsterdam, [5]Università di Trieste

June 2022

Outline

Introduction and preliminaries

DCP and Shortest $\mathcal{S}$-Equivalent String

Combinatorial bounds

Solving z-Shortest $\mathcal{S}$-Equivalent String

Shortest equivalent string : problem definition

SHORTEST $\mathcal{S}$-EQUIVALENT STRING
Input: A set $\mathcal{S}$ of n length-k strings.
Output: A shortest string T such that the set of length-k substrings of T is $\mathcal{S}$, or FAIL if that is not possible.

Shortest equivalent string : problem definition

SHORTEST $\mathcal{S}$-EQUIVALENT STRING
Input: A set $\mathcal{S}$ of n length-k strings.
Output: A shortest string T such that the set of length-k substrings of T is $\mathcal{S}$, or FAIL if that is not possible.

Example : $\mathcal{S} = \{\texttt{abr}, \texttt{bra}, \texttt{rac}, \texttt{aca}, \texttt{cad}, \texttt{ada}, \texttt{dab}\}$

Shortest equivalent string : problem definition

SHORTEST $\mathcal{S}$-EQUIVALENT STRING
Input: A set $\mathcal{S}$ of n length-k strings.
Output: A shortest string T such that the set of length-k substrings of T is $\mathcal{S}$, or FAIL if that is not possible.

Example : $\mathcal{S} = \{\texttt{abr}, \texttt{bra}, \texttt{rac}, \texttt{aca}, \texttt{cad}, \texttt{ada}, \texttt{dab}\}$
The string `abracadabra` has $\mathcal{S}$ for set of length 3 substrings.

Shortest equivalent string : problem definition

Shortest $\mathcal{S}$-Equivalent String
Input: A set $\mathcal{S}$ of n length-k strings.
Output: A shortest string T such that the set of length-k substrings of T is $\mathcal{S}$, or FAIL if that is not possible.

Example : $\mathcal{S} = \{\texttt{abr}, \texttt{bra}, \texttt{rac}, \texttt{aca}, \texttt{cad}, \texttt{ada}, \texttt{dab}\}$
The string `abracadabra` has $\mathcal{S}$ for set of length 3 substrings.
The strings `abracadab` or `cadabraca` (for example) are shortest for this property.

Shortest equivalent string : problem definition

SHORTEST $\mathcal{S}$-EQUIVALENT STRING
Input: A set $\mathcal{S}$ of n length-k strings.
Output: A shortest string T such that the set of length-k substrings of T is $\mathcal{S}$, or FAIL if that is not possible.

Example : $\mathcal{S} = \{\texttt{abr}, \texttt{bra}, \texttt{rac}, \texttt{aca}, \texttt{cad}, \texttt{ada}, \texttt{dab}\}$
The string abracadabra has $\mathcal{S}$ for set of length 3 substrings.
The strings abracadab or cadabraca (for example) are shortest for this property.

Generalization z-SHORTEST $\mathcal{S}$-EQUIVALENT STRING: We want the z shortest strings satisfying the property, in increasing length order.

Motivation

- Data privacy

Motivation

- Data privacy
  - Private string : `abracadabra`.

Motivation

- Data privacy
  - Private string : `abracadabra`.
  - Public string for pattern matching queries : `cadabraca`.

Motivation

- Data privacy
  - Private string : `abracadabra`.
  - Public string for pattern matching queries : `cadabraca`.
  - One can answer pattern matching queries for patterns shorter than 3.

Motivation

- Data privacy
  - Private string : `abracadabra`.
  - Public string for pattern matching queries : `cadabraca`.
  - One can answer pattern matching queries for patterns shorter than 3.
- Data compression

Motivation

- Data privacy
  - Private string : `abracadabra`.
  - Public string for pattern matching queries : `cadabraca`.
  - One can answer pattern matching queries for patterns shorter than 3.
- Data compression
- Bioinformatics

Motivation

- Data privacy
  - Private string : `abracadabra`.
  - Public string for pattern matching queries : `cadabraca`.
  - One can answer pattern matching queries for patterns shorter than 3.
- Data compression
- Bioinformatics
  - The length k strings are k-mers in a genome.

Motivation

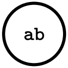
- Data privacy
  - Private string : `abracadabra`.
  - Public string for pattern matching queries : `cadabraca`.
  - One can answer pattern matching queries for patterns shorter than 3.
- Data compression
- Bioinformatics
  - The length k strings are k-mers in a genome.
  - Find a shorter string having a given *k-mer spectrum*.

De Bruijn graph of a set of length-k strings

$\mathcal{S} = \{\mathtt{abr}, \mathtt{bra}, \mathtt{rac}, \mathtt{aca}, \mathtt{cad}, \mathtt{ada}, \mathtt{dab}\}$

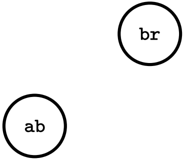
De Bruijn graph of a set of length-k strings

$\mathcal{S} = \{\underline{\mathtt{ab}}\mathtt{r}, \mathtt{bra}, \mathtt{rac}, \mathtt{aca}, \mathtt{cad}, \mathtt{ada}, \mathtt{dab}\}$



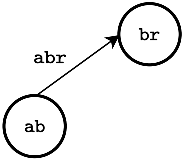
De Bruijn graph of a set of length-k strings

$\mathcal{S} = \{\mathtt{a}\underline{\mathtt{br}}, \mathtt{bra}, \mathtt{rac}, \mathtt{aca}, \mathtt{cad}, \mathtt{ada}, \mathtt{dab}\}$



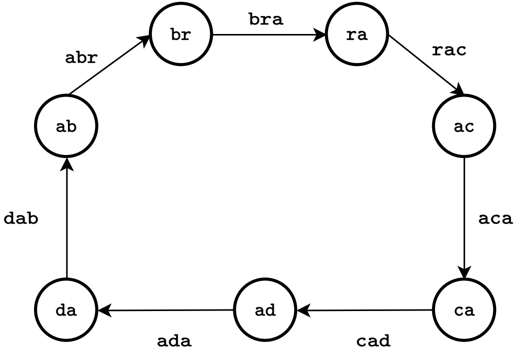
De Bruijn graph of a set of length-k strings

$\mathcal{S} = \{\mathtt{abr}, \mathtt{bra}, \mathtt{rac}, \mathtt{aca}, \mathtt{cad}, \mathtt{ada}, \mathtt{dab}\}$



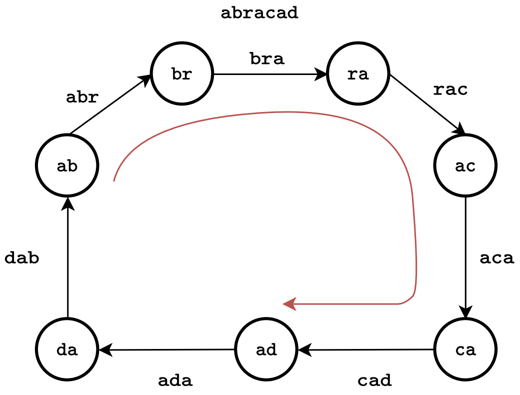
De Bruijn graph of a set of length-k strings

$\mathcal{S} = \{\texttt{abr}, \texttt{bra}, \texttt{rac}, \texttt{aca}, \texttt{cad}, \texttt{ada}, \texttt{dab}\}$



De Bruijn graph of a set of length-k strings

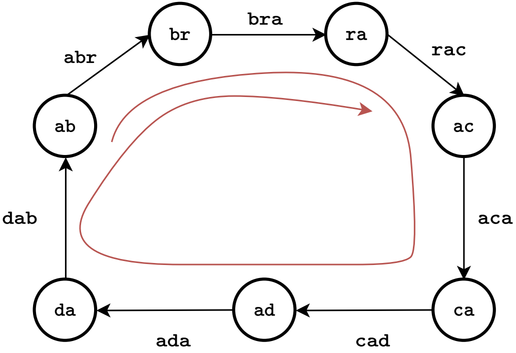
$\mathcal{S} = \{\texttt{abr}, \texttt{bra}, \texttt{rac}, \texttt{aca}, \texttt{cad}, \texttt{ada}, \texttt{dab}\}$



De Bruijn graph of a set of length-k strings

$\mathcal{S} = \{\texttt{abr}, \texttt{bra}, \texttt{rac}, \texttt{aca}, \texttt{cad}, \texttt{ada}, \texttt{dab}\}$

abracadabra



Eulerian walks

Proposition

The strings having the set of their length-k substrings *included in* $\mathcal{S}$ correspond to walks on the de Bruijn graph of $\mathcal{S}$.

Eulerian walks

Proposition

The strings having the set of their length-k substrings *included in* $\mathcal{S}$ correspond to walks on the de Bruijn graph of $\mathcal{S}$.

Definition

An *Eulerian walk* on a graph G is a walk on G that traverses every edge *at least* once.

Eulerian walks

Proposition

The strings having the set of their length-k substrings *equal to* $\mathcal{S}$ correspond to *Eulerian* walks on the de Bruijn graph of $\mathcal{S}$.

Definition

An *Eulerian walk* on a graph G is a walk on G that traverses every edge *at least* once.

Eulerian walks

Consequence : To solve Shortest $\mathcal{S}$-Equivalent String and z-Shortest $\mathcal{S}$-Equivalent String, we need to find Eulerian walks in De Bruijn graphs.

Outline

Introduction and preliminaries

DCP and SHORTEST $\mathcal{S}$-EQUIVALENT STRING

Combinatorial bounds

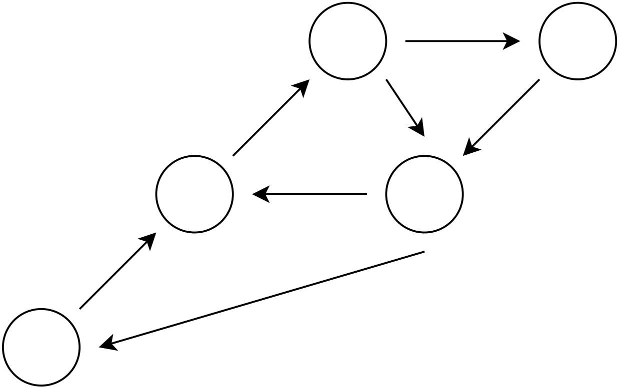
Solving z-SHORTEST $\mathcal{S}$-EQUIVALENT STRING

Directed Chinese Postman problem

Directed Chinese Postman (DCP)
Input: A directed graph $G(V, E)$.
Output: A shortest closed Eulerian walk, or FAIL if that is not possible.

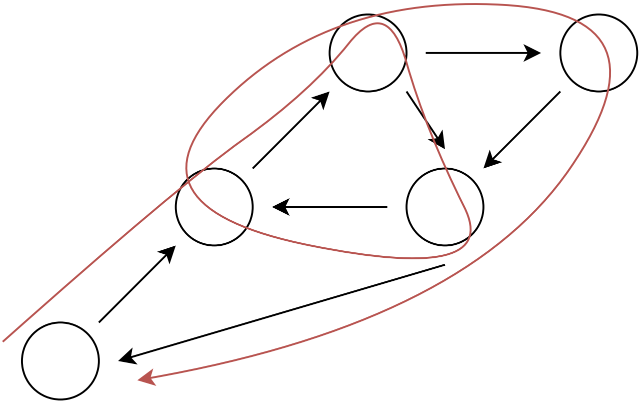
Directed Chinese Postman problem

Directed Chinese Postman (DCP)
Input: A directed graph $G(V, E)$.
Output: A shortest closed Eulerian walk, or FAIL if that is not possible.



Directed Chinese Postman problem

DIRECTED CHINESE POSTMAN (DCP)
Input: A directed graph $G(V, E)$.
Output: A shortest closed Eulerian walk, or FAIL if that is not possible.

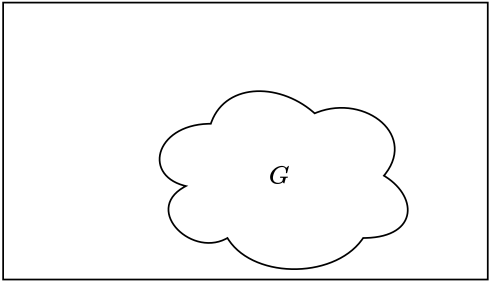


Opened and closed Eulerian walks

We can ignore the requirement of walks being closed via the following trick :

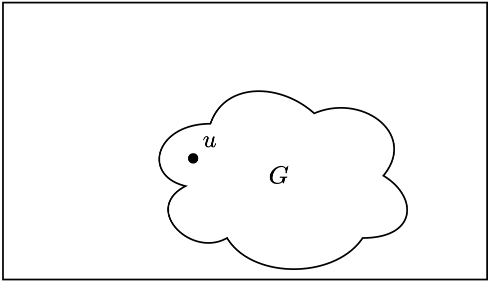
Opened and closed Eulerian walks

We can ignore the requirement of walks being closed via the following trick :



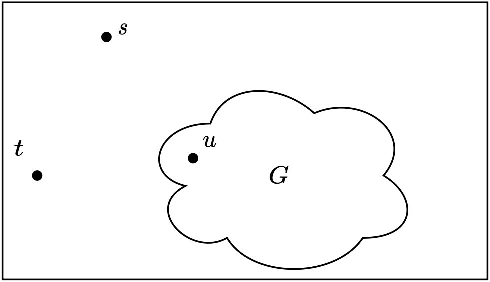
Opened and closed Eulerian walks

We can ignore the requirement of walks being closed via the following trick :



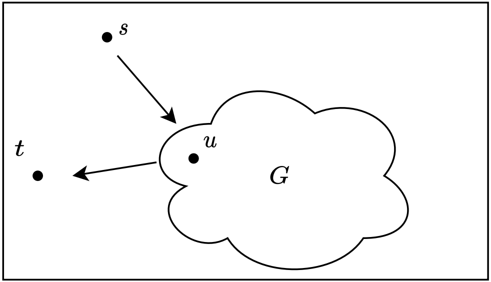
Opened and closed Eulerian walks

We can ignore the requirement of walks being closed via the following trick :



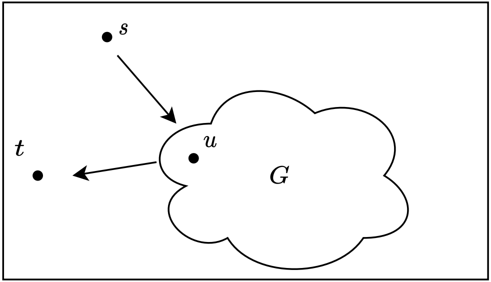
Opened and closed Eulerian walks

We can ignore the requirement of walks being closed via the following trick :



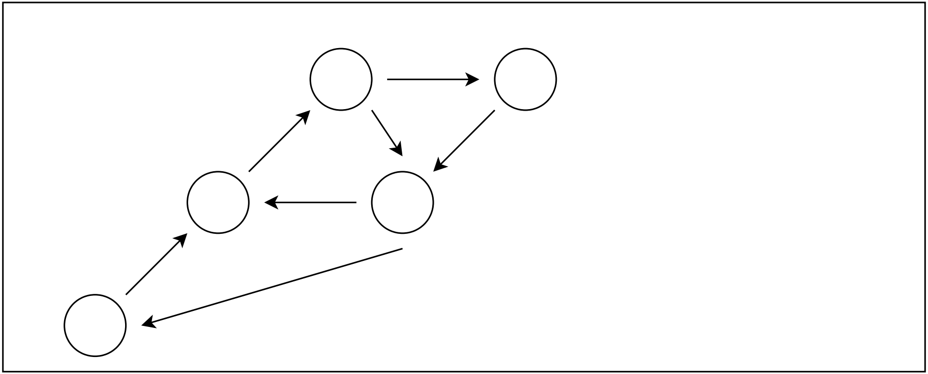
Opened and closed Eulerian walks

We can ignore the requirement of walks being closed via the following trick :

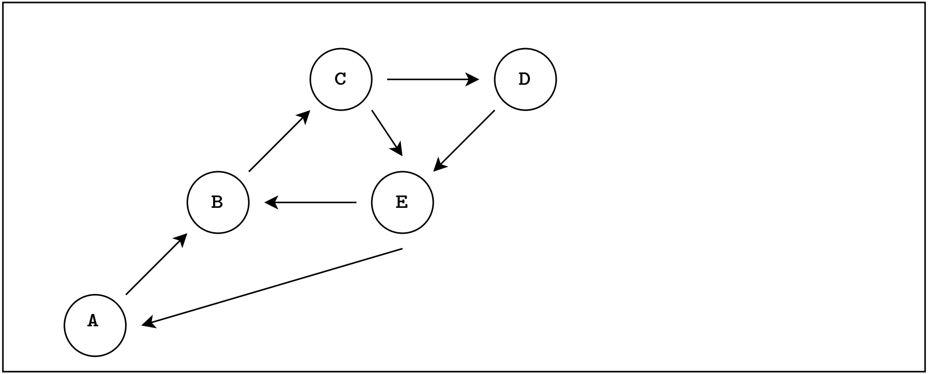


One can find the closed Eulerian walks on G by finding the open Eulerian walks on the extended graph.

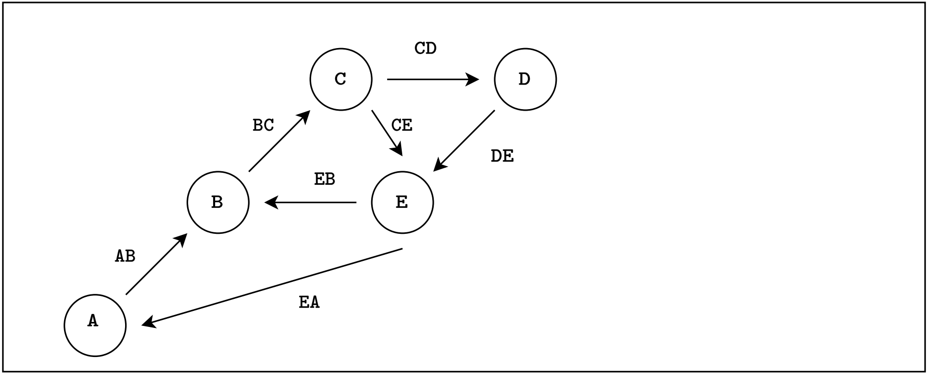
Reducing DIRECTED CHINESE POSTMAN to SHORTEST $\mathcal{S}$-EQUIVALENT STRING (large alphabet)



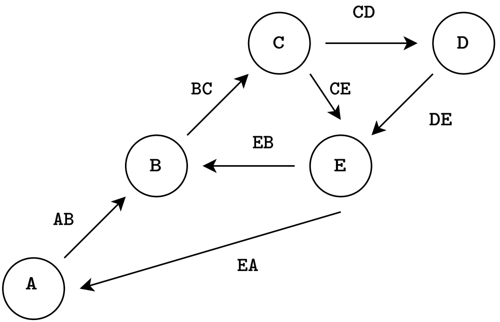
Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (large alphabet)



Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (large alphabet)

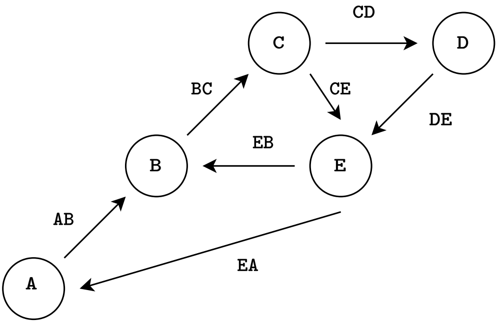


Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (large alphabet)



$$\left\{ \begin{array}{c} \mathtt{AB}, \mathtt{EA}, \\ \mathtt{EB}, \mathtt{BC}, \\ \mathtt{CD}, \mathtt{DE} \\ \mathtt{CE} \end{array} \right\}$$

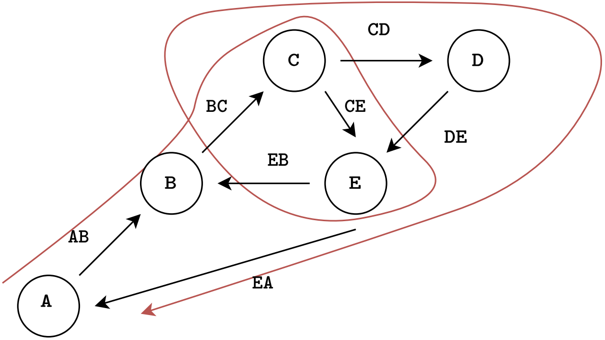
Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (large alphabet)



$$\left\{ \begin{array}{c} \mathtt{AB}, \mathtt{EA}, \\ \mathtt{EB}, \mathtt{BC}, \\ \mathtt{CD}, \mathtt{DE} \\ \mathtt{CE} \end{array} \right\}$$

ABCEBCDEA

Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (large alphabet)



$$\left\{ \begin{array}{c} \texttt{AB}, \texttt{EA}, \\ \texttt{EB}, \texttt{BC}, \\ \texttt{CD}, \texttt{DE} \\ \texttt{CE} \end{array} \right\}$$

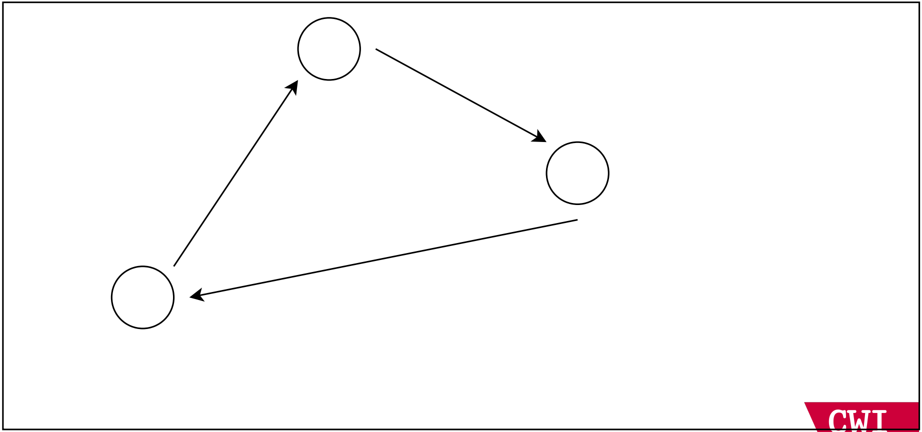
ABCEBCDEA

Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (large alphabet)

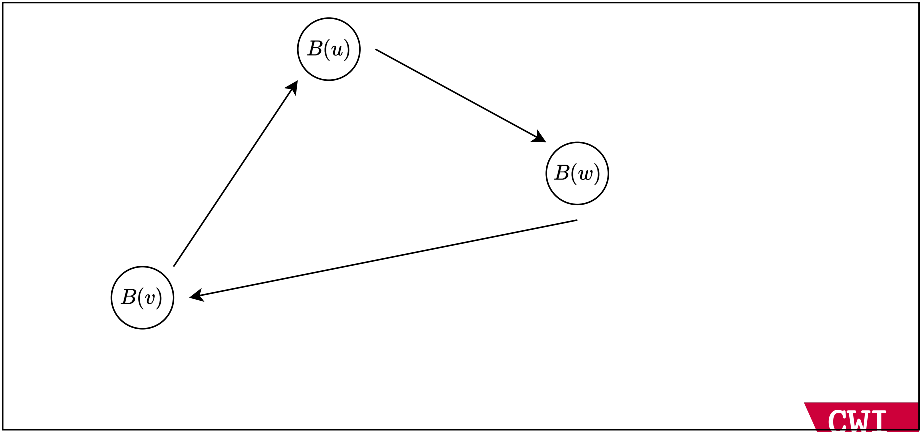
Theorem

Any instance of Directed Chinese Postman *can be reduced to an instance of* Shortest $\mathcal{S}$-Equivalent String *in linear time with* $||\mathcal{S}|| = \mathcal{O}(|E|)$.

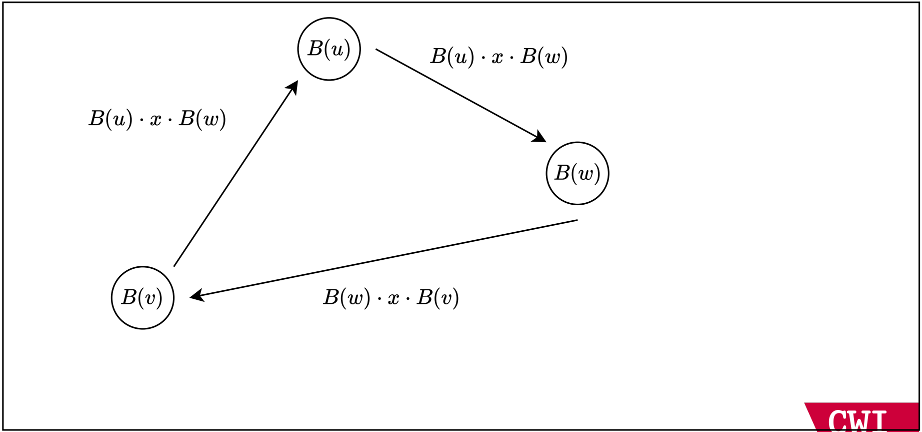
Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (small alphabet)



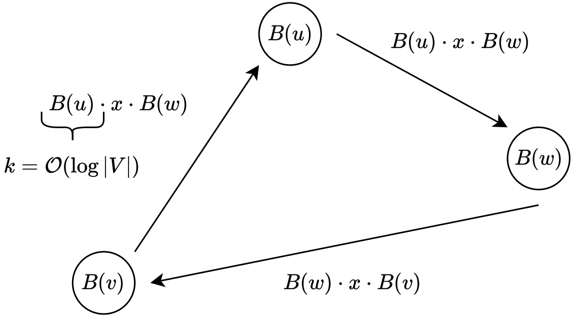
Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (small alphabet)



Reducing DIRECTED CHINESE POSTMAN to SHORTEST $\mathcal{S}$-EQUIVALENT STRING (small alphabet)

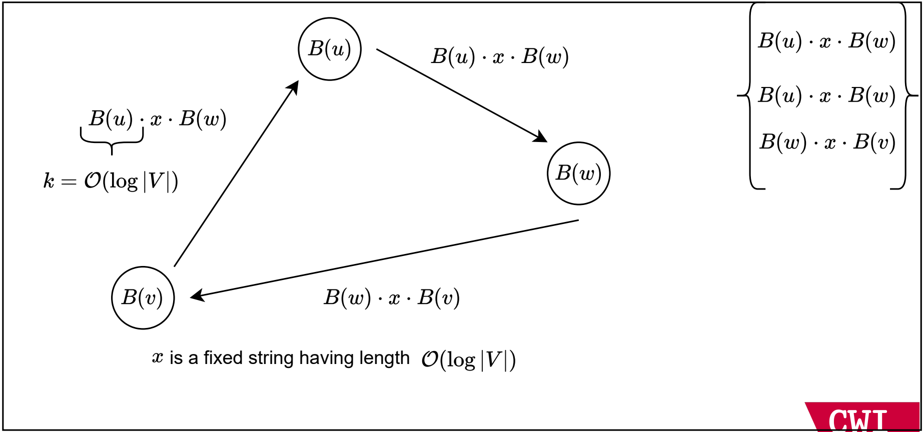


Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (small alphabet)

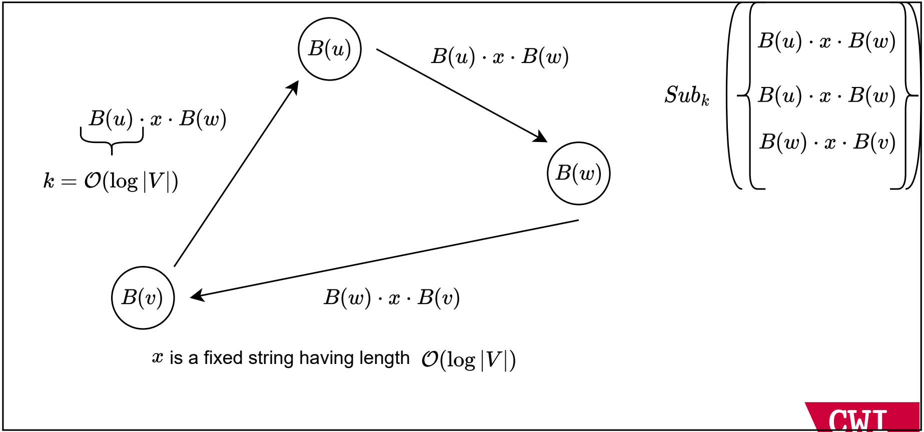


x is a fixed string having length $\mathcal{O}(\log |V|)$

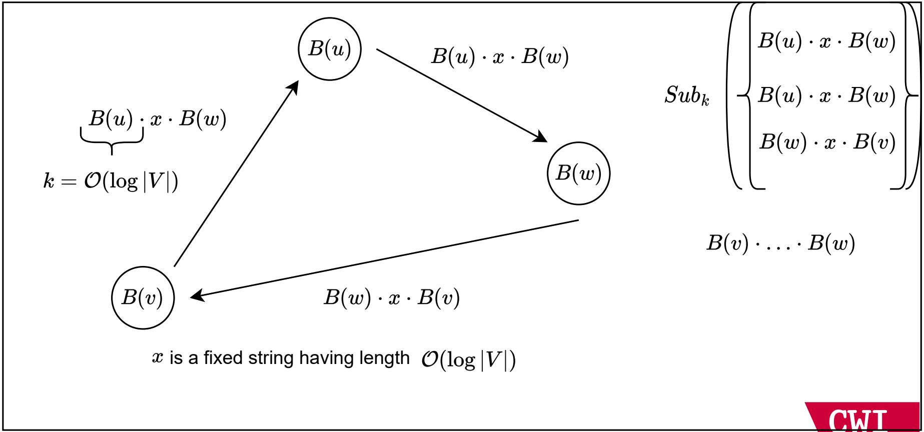
Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (small alphabet)



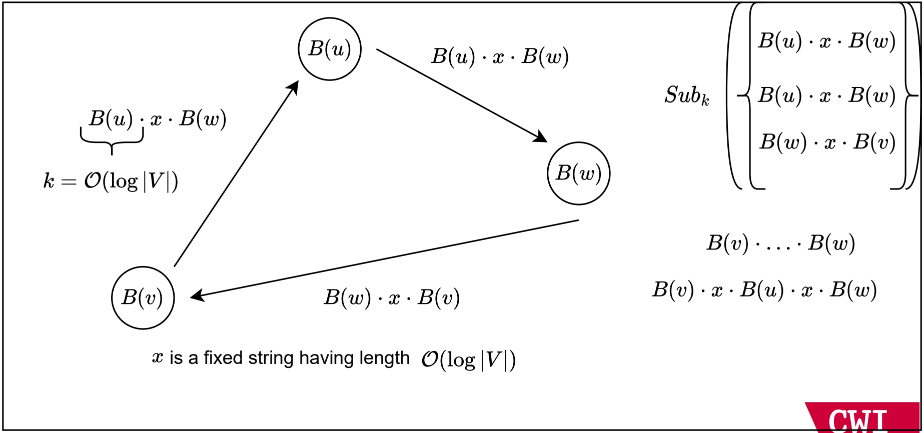
Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (small alphabet)



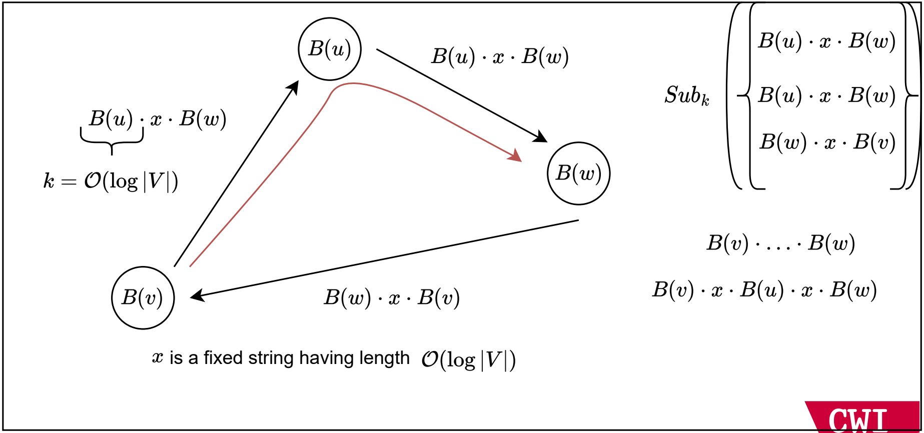
Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (small alphabet)



Reducing DIRECTED CHINESE POSTMAN to SHORTEST $\mathcal{S}$-EQUIVALENT STRING (small alphabet)



Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (small alphabet)



Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (small alphabet)

Theorem

Any instance $G(V, E)$ of Directed Chinese Postman *with output $\mathcal{W}$ can be reduced in $\mathcal{O}(|E| \log |V| + |\mathcal{W}|)$ time to an instance of* Shortest $\mathcal{S}$-Equivalent String *on a binary alphabet, where $\mathcal{S}$ is a set of $\mathcal{O}(|E| \log |V|)$ strings having length $\log |V|$.*

Reducing Directed Chinese Postman to Shortest $\mathcal{S}$-Equivalent String (small alphabet)

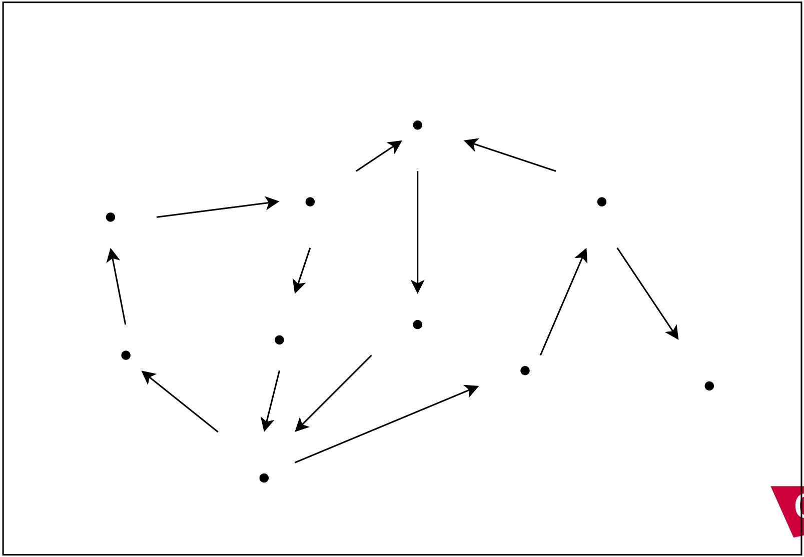
Theorem

Any instance $G(V, E)$ of Directed Chinese Postman *with output $\mathcal{W}$ can be reduced in $\mathcal{O}(|E| \log |V| + |\mathcal{W}|)$ time to an instance of* Shortest $\mathcal{S}$-Equivalent String *on a binary alphabet, where $\mathcal{S}$ is a set of $\mathcal{O}(|E| \log |V|)$ strings having length $\log |V|$.*

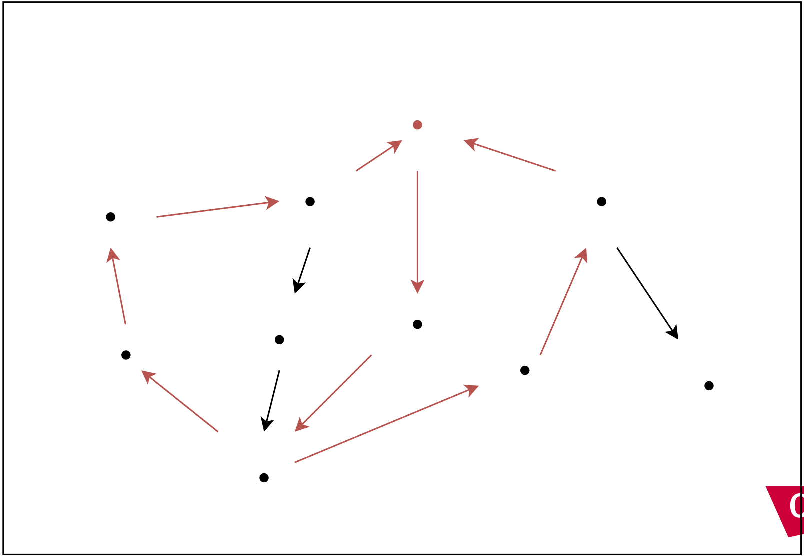
Consequence : If the Shortest $\mathcal{S}$-Equivalent String problem over a *binary alphabet* has a near-linear-time solution then so does Directed Chinese Postman.

Outline

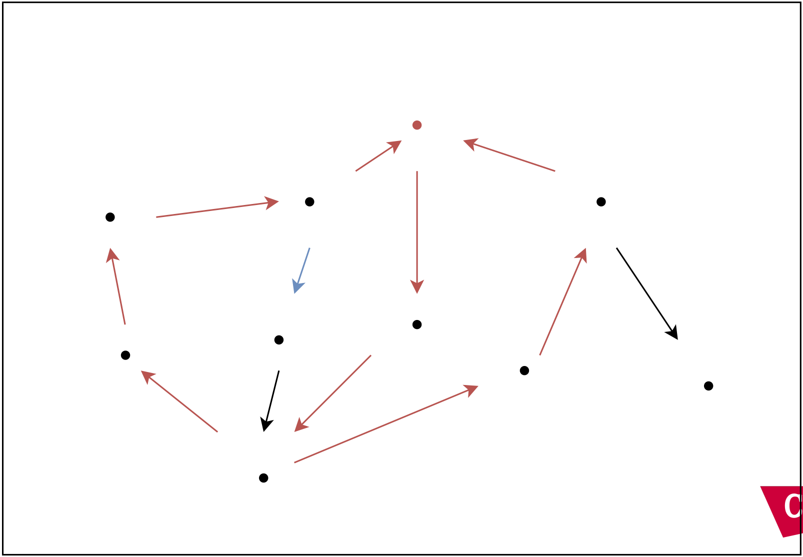
Length of the shortest Eulerian walk on a graph



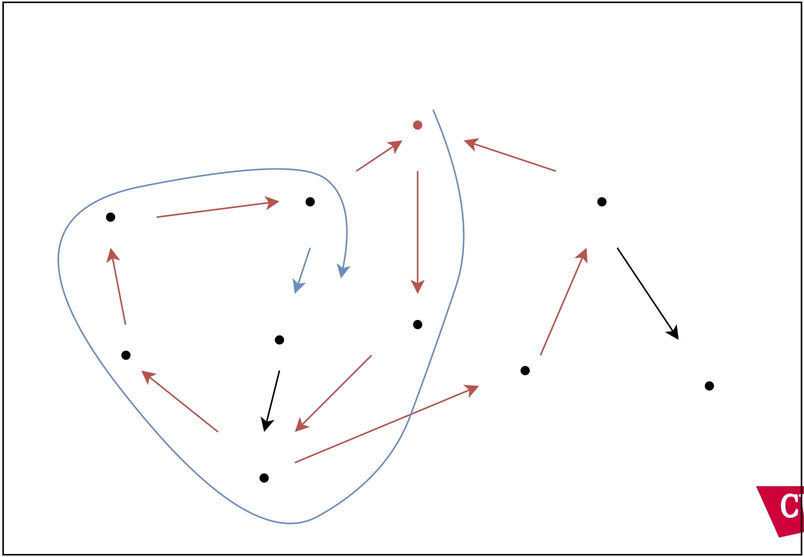
Length of the shortest Eulerian walk on a graph



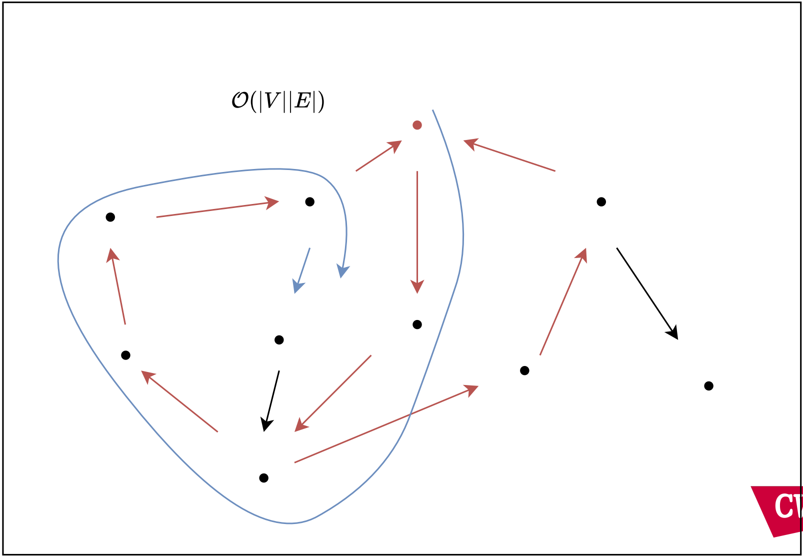
Length of the shortest Eulerian walk on a graph



Length of the shortest Eulerian walk on a graph



Length of the shortest Eulerian walk on a graph



Length of the shortest Eulerian walk on a graph

Theorem

If $\mathcal{W}$ is the shortest Eulerian walk on a graph $G(V, E)$, then $|\mathcal{W}| = \mathcal{O}(|V|\,|E|)$.

Length of the shortest Eulerian walk on a graph

Theorem

If $\mathcal{W}$ is the shortest Eulerian walk on a graph $G(V, E)$, then $|\mathcal{W}| = \mathcal{O}(|V|\,|E|)$ and this bound is asymptotically tight.

Length of the shortest Eulerian walk on a graph

Theorem

If $\mathcal{W}$ is the shortest Eulerian walk on a graph $G(V, E)$, then $|\mathcal{W}| = \mathcal{O}(|V|\,|E|)$ and this bound is asymptotically tight.

Lemma

Let $\mathcal{S}$ be a set containing n strings of length k each. Then the De Bruijn graph of $\mathcal{S}$ has at most $n+1$ nodes and n edges. In this graph, a walk $\mathcal{W}$ corresponds to a string of length $|\mathcal{W}| + k - 1$.

Length of the shortest Eulerian walk on a graph

Theorem

If $\mathcal{W}$ is the shortest Eulerian walk on a graph $G(V, E)$, then $|\mathcal{W}| = \mathcal{O}(|V|\,|E|)$ and this bound is asymptotically tight.

Lemma

Let $\mathcal{S}$ be a set containing n strings of length k each. Then the De Bruijn graph of $\mathcal{S}$ has at most $n+1$ nodes and n edges. In this graph, a walk $\mathcal{W}$ corresponds to a string of length $|\mathcal{W}| + k - 1$.

Theorem

If $\mathcal{T}$ is the output of SHORTEST $\mathcal{S}$-EQUIVALENT STRING, *then $|\mathcal{T}| = \mathcal{O}(k + n^2)$.*

Total length of the z shortest Eulerian walks on a graph

- If the graph is acyclic, there is at most one walk.

Total length of the z shortest Eulerian walks on a graph

- If the graph is acyclic, there is at most one walk.
- Otherwise we can traverse the shortest cycle once more to get a new walk.

Total length of the z shortest Eulerian walks on a graph

- If the graph is acyclic, there is at most one walk.
- Otherwise we can traverse the shortest cycle once more to get a new walk.
- The shortest cycle has length smaller than $|V|$.

Total length of the z shortest Eulerian walks on a graph

- If the graph is acyclic, there is at most one walk.
- Otherwise we can traverse the shortest cycle once more to get a new walk.
- The shortest cycle has length smaller than $|V|$.
- We obtain total length $\mathcal{O}(\sum_{i=0}^{z}(|V||E| + i|V|))$.

Total length of the z shortest Eulerian walks on a graph

- If the graph is acyclic, there is at most one walk.
- Otherwise we can traverse the shortest cycle once more to get a new walk.
- The shortest cycle has length smaller than $|V|$.
- We obtain total length $\mathcal{O}(\sum_{i=0}^{z}(|V||E| + i|V|)) = \mathcal{O}(z|V||E| + z^2|V|)$.

Total length of the z shortest Eulerian walks on a graph

Theorem

If $||\mathcal{W}_z||$ is the cumulative length of the z shortest Eulerian walks on a graph $G(V, E)$, then $||\mathcal{W}_z|| = \mathcal{O}(z|V|\,|E| + z^2|V|)$ and this bound is asymptotically tight.

Total length of the z shortest Eulerian walks on a graph

Theorem

If $||\mathcal{W}_z||$ is the cumulative length of the z shortest Eulerian walks on a graph $G(V,E)$, then $||\mathcal{W}_z|| = \mathcal{O}(z|V|\,|E| + z^2|V|)$ and this bound is asymptotically tight.

Theorem

*If $\mathcal{T}_z$ is the output of z-*Shortest $\mathcal{S}$-Equivalent String *for n strings of length k each, then $||\mathcal{T}_z|| = \mathcal{O}(zk + zn^2 + z^2 n)$.*

Outline

Introduction and preliminaries

DCP and SHORTEST $\mathcal{S}$-EQUIVALENT STRING

Combinatorial bounds

Solving z-SHORTEST $\mathcal{S}$-EQUIVALENT STRING

Definition : semi-Eulerian graph

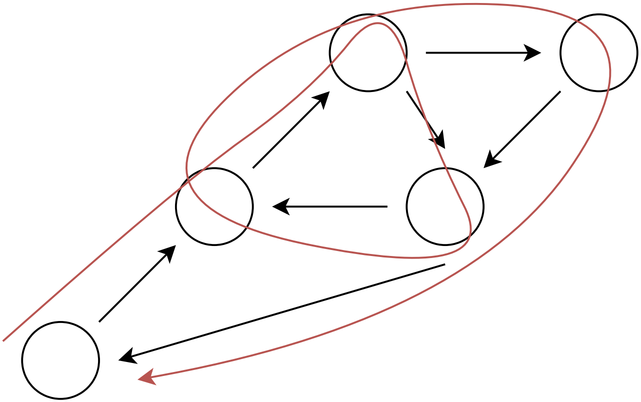
Definition

We say that an Eulerian walk is an *Eulerian trail* if it visits every edge *exactly* once.

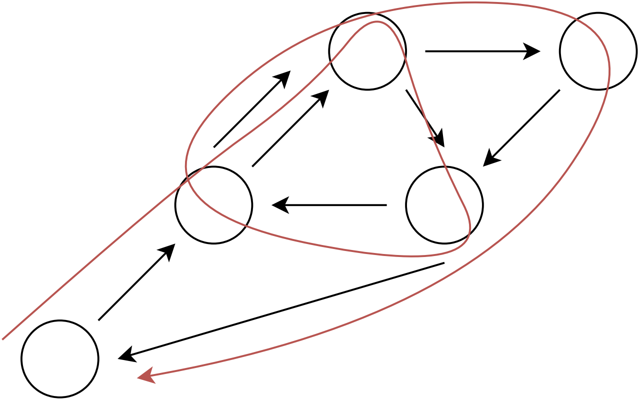
Definition

We say that a graph G is *semi-Eulerian* if there is an Eulerian trail on G.

Semi-Eulerian extensions



Semi-Eulerian extensions



Solving Shortest $\mathcal{S}$-Equivalent String via Eulerian walks

Reminder : To solve Shortest $\mathcal{S}$-Equivalent String and z-Shortest $\mathcal{S}$-Equivalent String, we need to find Eulerian walks in De Bruijn graphs.

Solving SHORTEST $\mathcal{S}$-EQUIVALENT STRING via Eulerian walks

Reminder : To solve SHORTEST $\mathcal{S}$-EQUIVALENT STRING and z-SHORTEST $\mathcal{S}$-EQUIVALENT STRING, we need to find Eulerian walks in De Bruijn graphs.

- We solve SHORTEST $\mathcal{S}$-EQUIVALENT STRING on a set $\mathcal{S}$ by :

Solving SHORTEST $\mathcal{S}$-EQUIVALENT STRING via Eulerian walks

Reminder : To solve SHORTEST $\mathcal{S}$-EQUIVALENT STRING and z-SHORTEST $\mathcal{S}$-EQUIVALENT STRING, we need to find Eulerian walks in De Bruijn graphs.

- We solve SHORTEST $\mathcal{S}$-EQUIVALENT STRING on a set $\mathcal{S}$ by :
  - Finding a minimal set of edge that can be copied to make the de Bruijn graph of $\mathcal{S}$ semi-Eulerian : we call *semi-Eulerian extension* such a graph with copied edges.

Solving SHORTEST $\mathcal{S}$-EQUIVALENT STRING via Eulerian walks

Reminder : To solve SHORTEST $\mathcal{S}$-EQUIVALENT STRING and z-SHORTEST $\mathcal{S}$-EQUIVALENT STRING, we need to find Eulerian walks in De Bruijn graphs.

- We solve SHORTEST $\mathcal{S}$-EQUIVALENT STRING on a set $\mathcal{S}$ by :
  - Finding a minimal set of edge that can be copied to make the de Bruijn graph of $\mathcal{S}$ semi-Eulerian : we call *semi-Eulerian extension* such a graph with copied edges.
  - Find Eulerian trails in the extended graph.

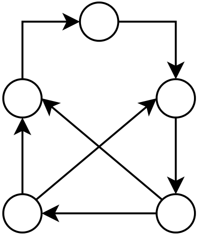
Solving Shortest $\mathcal{S}$-Equivalent String via Eulerian walks

Reminder : To solve Shortest $\mathcal{S}$-Equivalent String and z-Shortest $\mathcal{S}$-Equivalent String, we need to find Eulerian walks in De Bruijn graphs.

- We solve Shortest $\mathcal{S}$-Equivalent String on a set $\mathcal{S}$ by :
  - Finding a minimal set of edge that can be copied to make the de Bruijn graph of $\mathcal{S}$ semi-Eulerian : we call *semi-Eulerian extension* such a graph with copied edges.
  - Find Eulerian trails in the extended graph.
  - Bring back those trails in the original graph to find an Eulerian walk and therefore a $\mathcal{S}$ equivalent string.

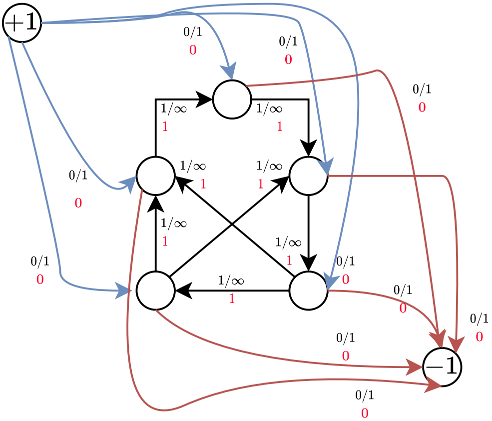
CWI

Finding Eulerian walks through flows



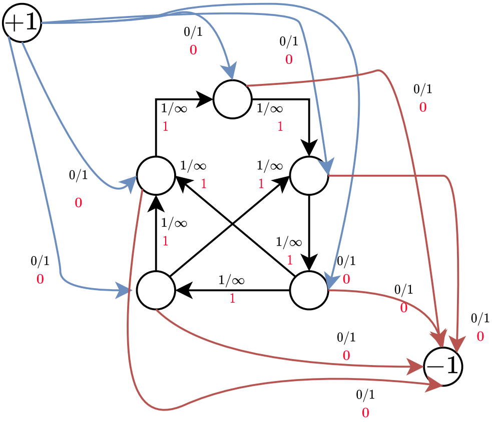
Now we want to model our extension problem with a flow problem.

Finding Eulerian walks through flows



Now we want to model our extension problem with a flow problem.
We extend the graph and create an instance of MinCostFlow.

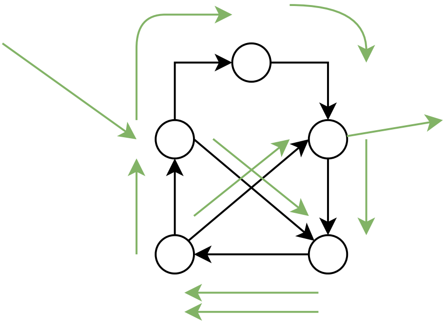
Finding Eulerian walks through flows



Now we want to model our extension problem with a flow problem.
We extend the graph and create an instance of MINCOSTFLOW.
Each feasible flow on this problem corresponds to a semi-Eulerian extension.

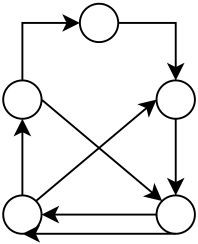
Link between flows on G_{ext} and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{\mathcal{O}}(|E||V|)$ ([Orlin, 1997],[Tarjan, 1997]).



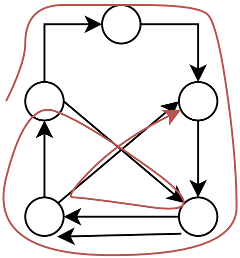
Link between flows on G_{ext} and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{\mathcal{O}}(|E||V|)$ ([Orlin, 1997],[Tarjan, 1997]).
- A flow gives us a semi-Eulerian extension of G obtained by copying the edges as many times as they are traversed .



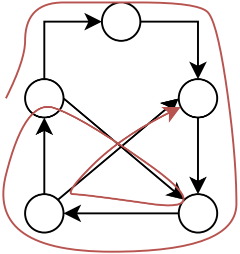
Link between flows on G_{ext} and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{\mathcal{O}}(|E||V|)$ ([Orlin, 1997],[Tarjan, 1997]).
- A flow gives us a semi-Eulerian extension of G obtained by copying the edges as many times as they are traversed .
- Find an Eulerian trail $\mathcal{W}$ on the extended graph (in $\mathcal{O}(|\mathcal{W}|)$).



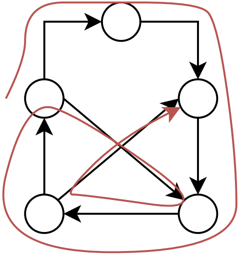
Link between flows on G_{ext} and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{\mathcal{O}}(|E||V|)$ ([Orlin, 1997],[Tarjan, 1997]).
- A flow gives us a semi-Eulerian extension of G obtained by copying the edges as many times as they are traversed .
- Find an Eulerian trail $\mathcal{W}$ on the extended graph (in $\mathcal{O}(|\mathcal{W}|)$).
- This Eulerian trail corresponds to an Eulerian walk on G.



Link between flows on G_{ext} and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{\mathcal{O}}(|E||V|)$ ([Orlin, 1997],[Tarjan, 1997]).
- A flow gives us a semi-Eulerian extension of G obtained by copying the edges as many times as they are traversed .
- Find an Eulerian trail $\mathcal{W}$ on the extended graph (in $\mathcal{O}(|\mathcal{W}|)$).
- This Eulerian trail corresponds to an Eulerian walk on G.
- A minimal cost flow gives us a shortest walk.



Results for Shortest $\mathcal{S}$-Equivalent String

Theorem

The Shortest $\mathcal{S}$-Equivalent String *problem can be solved in* $\tilde{\mathcal{O}}(nk + n^2)$ *time.*

Solving z-Shortest $\mathcal{S}$-Equivalent String

- z-Shortest $\mathcal{S}$-Equivalent String: One wants the z shortest strings.

Solving z-Shortest $\mathcal{S}$-Equivalent String

- z-Shortest $\mathcal{S}$-Equivalent String: One wants the z shortest strings.
- Finding z minimal cost flows can be done in $\tilde{\mathcal{O}}(z|V|^3)$ time (dense graphs), or in $\tilde{\mathcal{O}}(z(|E||V| + |V|^2))$ time (sparse graphs) ([Könen et al., 2021]).

Solving z-Shortest $\mathcal{S}$-Equivalent String

- z-Shortest $\mathcal{S}$-Equivalent String: One wants the z shortest strings.
- Finding z minimal cost flows can be done in $\tilde{\mathcal{O}}(z|V|^3)$ time (dense graphs), or in $\tilde{\mathcal{O}}(z(|E||V| + |V|^2))$ time (sparse graphs) ([Könen et al., 2021]).
- In a semi Eulerian graph, one can compute the set $\mathcal{W}_z$ of the z smallest Eulerian walks in linear time ([Conte et al., 2021] or [Kurita and Wasa, 2021]).

CWI

Solving z-Shortest $\mathcal{S}$-Equivalent String

- z-Shortest $\mathcal{S}$-Equivalent String: One wants the z shortest strings.
- Finding z minimal cost flows can be done in $\tilde{\mathcal{O}}(z|V|^3)$ time (dense graphs), or in $\tilde{\mathcal{O}}(z(|E||V|+|V|^2))$ time (sparse graphs) ([Könen et al., 2021]).
- In a semi Eulerian graph, one can compute the set $\mathcal{W}_z$ of the z smallest Eulerian walks in linear time ([Conte et al., 2021] or [Kurita and Wasa, 2021]).
- As before, each flow corresponds at least to an Eulerian walk and the z minimal cost flows correspond to at least z Eulerian walks.

Solving z-Shortest $\mathcal{S}$-Equivalent String

- z-Shortest $\mathcal{S}$-Equivalent String: One wants the z shortest strings.
- Finding z minimal cost flows can be done in $\tilde{\mathcal{O}}(z|V|^3)$ time (dense graphs), or in $\tilde{\mathcal{O}}(z(|E||V| + |V|^2))$ time (sparse graphs) ([Könen et al., 2021]).
- In a semi Eulerian graph, one can compute the set $\mathcal{W}_z$ of the z smallest Eulerian walks in linear time ([Conte et al., 2021] or [Kurita and Wasa, 2021]).
- As before, each flow corresponds at least to an Eulerian walk and the z minimal cost flows correspond to at least z Eulerian walks.

Theorem

The z-Shortest $\mathcal{S}$-Equivalent String problem can be solved in $\tilde{\mathcal{O}}(nk + zn^2 + ||\mathcal{T}_z||)$ time.

Solving z-Shortest $\mathcal{S}$-Equivalent String

- z-Shortest $\mathcal{S}$-Equivalent String: One wants the z shortest strings.
- Finding z minimal cost flows can be done in $\tilde{\mathcal{O}}(z|V|^3)$ time (dense graphs), or in $\tilde{\mathcal{O}}(z(|E||V|+|V|^2))$ time (sparse graphs) ([Könen et al., 2021]).
- In a semi Eulerian graph, one can compute the set $\mathcal{W}_z$ of the z smallest Eulerian walks in linear time ([Conte et al., 2021] or [Kurita and Wasa, 2021]).
- As before, each flow corresponds at least to an Eulerian walk and the z minimal cost flows correspond to at least z Eulerian walks.

Theorem

The z-Shortest $\mathcal{S}$-Equivalent String problem can be solved in $\tilde{\mathcal{O}}(nk + zn^2 + ||\mathcal{T}_z||)=\tilde{\mathcal{O}}(nk + zn^2 + zk)$ time.

Thanks for your attention !

This project is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska- Curie grant agreement No 956229.

Co-financed by the Connecting Europe Facility of the European Union

References I

Conte, A., Grossi, R., Loukides, G., Pisanti, N., Pissis, S. P., and Punzi, G. (2021).
Beyond the BEST theorem: Fast assessment of Eulerian trails.
In *Fundamentals of Computation Theory - 23rd International Symposium,*, volume 12867 of *Lecture Notes in Computer Science*, pages 162–175. Springer.

Könen, D., Schmidt, D. R., and Spisla, C. (2021).
Finding all minimum cost flows and a faster algorithm for the K best flow problem.
CoRR, abs/2105.10225.

Kurita, K. and Wasa, K. (2021).
Constant amortized time enumeration of Eulerian trails.
CoRR, abs/2101.10473.

References II

Orlin, J. B. (1997).
A polynomial time primal network simplex algorithm for minimum cost flows.
Math. Program., 77:109–129.

Tarjan, R. E. (1997).
Dynamic trees as search trees via Euler tours, applied to the network simplex algorithm.
Math. Program., 77:169–177.