On Strings Having the Same Length-k Substrings

Dutch Optimization Seminar - PhD Talk

Giulia Bernardini\textsuperscript{1}, Alessio Conte\textsuperscript{2}, Estéban Gabory\textsuperscript{1}, Roberto Grossi\textsuperscript{2}, Grigorios Loukides\textsuperscript{3}, Solon P. Pissis\textsuperscript{1,4}, Giulia Punzi\textsuperscript{2} and Michelle Sweering\textsuperscript{1}

\textsuperscript{1}CWI Amsterdam, \textsuperscript{2}Università di Pisa, \textsuperscript{3}King’s College London, \textsuperscript{4}Vrije Universiteit Amsterdam

March 31\textsuperscript{st}, 2022
Shortest equivalent string: problem definition

**Shortest S-Equivalent String**

**Input**: A set $S$ of $n$ length-$k$ strings

**Output**: A shortest string $T$ such that the set of $k$ substrings of $T$ is $S$, or FAIL if that is not possible.
Shortest $S$-Equivalent String

**Input:** A set $S$ of $n$ length-$k$ strings

**Output:** A shortest string $T$ such that the set of $k$ substrings of $T$ is $S$, or FAIL if that is not possible.

If $k = 3$, $S = \{”abr”, ”bra”, ”rac”, ”aca”, ”cad”, ”ada”, ”dab” \}$
Shortest equivalent string: problem definition

**Shortest S-Equivalent String**

**Input:** A set $S$ of $n$ length-$k$ strings  
**Output:** A shortest string $T$ such that the set of $k$ substrings of $T$ is $S$, or FAIL if that is not possible.

If $k = 3$, $S = \{"abr","bra","rac","aca","cad","ada","dab"\}$
The string "abracadabra" has $S$ for set of 3-substrings
Shortest equivalent string: problem definition

**Shortest S-Equivalent String**

**Input:** A set $S$ of $n$ length-$k$ strings

**Output:** A shortest string $T$ such that the set of $k$ substrings of $T$ is $S$, or FAIL if that is not possible.

If $k = 3$, $S = \{"abr","bra","rac","aca","cad","ada","dab"\}$

The string "abracadabra" has $S$ for set of 3-substrings.

The strings "abracadab" or "cadabraca" (for example) are shortest for this property.
Motivations

▶ Data privacy
Motivations

- Data privacy
  - Private string: "abracadabra".

- Data compression
- Bioinformatics
Motivations

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

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

- 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
Motivations

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

- **Data compression**

- **Bioinformatics**
De Bruijn graph of a string

Definition (De Bruijn graph of order \(k\) of a string \(T\))

- Nodes are the length \((k - 1)\) substrings of \(T\)
De Bruijn graph of a string

Definition (De Bruijn graph of order $k$ of a string $T$)

- Nodes are the length $(k - 1)$ substrings of $T$
- Each time a $k$ substring $u$ is in $T$, we add a directed edge from $u[0..k - 2]$ to $u[1..k - 1]$. 
Definition (De Bruijn graph of order $k$ of a string $T$)

- Nodes are the length $(k - 1)$ substrings of $T$
- Each time a $k$ substring $u$ is in $T$, we add a directed edge from $u[0..k - 2]$ to $u[1..k - 1]$.

Figure: De Bruijn graph of order 3 for $T = "abracadabra"$
Definition (De Bruijn graph of order $k$ of a string $T$)

- Nodes are the length $(k - 1)$ substrings of $T$
- Each time a $k$ substring $u$ is in $T$, we add a directed edge from $u[0..k-2]$ to $u[1..k-1]$.

Figure: De Bruijn graph of order 3 for $T = \text{"abracadabra"}$
De Bruijn graph of a string

**Definition**
A graph $G$ is *semi-Eulerian* if it can be traversed by visiting each edge once and only once.

**Figure:** De Bruijn graph of order 3 for $T = “abracadabra”$
Definition
A graph $G$ is semi-Eulerian if it can be traversed by visiting each edge once and only once.

Proposition
If a graph $G$ is a de Bruijn graph of a string, then $G$ is semi-Eulerian.

Figure: De Bruijn graph of order 3 for $T = \text{"abracadabra"}$
Eulerian trails

Definition
An *Eulerian trail* on a graph $G$ is a walk on $G$ that traverses every edges *exactly* once.
De Bruijn graph of a set of $k$ strings

$$S_k = \{"abr", "aca", "dab", "abc"\}$$  \hspace{1cm} k = 3

**Figure:** De Bruijn graph of order 3 for $S = \{"abr", "aca", "dab", "abc"\}$
De Bruijn graph of a set of $k$ strings

$k = 3$

$S_k = \{"abr","bra","rac","aca","cad","ada","dab"\}$

Figure: De Bruijn graph of order 3 for $S = \{"abr","bra","rac","aca","cad","ada","dab"\}$
De Bruijn graph of a set of $k$ strings

$k = 3$

$S_k = \{"abr","bra","rac","aca","cad","ada","dab"\}$

Figure: De Bruijn graph of order 3 for $S = \{"abr","bra","rac","aca","cad","ada","dab"\}$
An Eulerian walk on a graph $G$ is a walk on $G$ that traverses every edge at least once.
Proposition

The strings having $S$ as a set of $k$-substrings corresponds to walks traversing at least once each edge of the order $k$ de Bruijn graph of $S$. 
Proposition

The strings having $S$ as a set of $k$-substrings corresponds to walks traversing at least once each edge of the order $k$ de Bruijn graph of $S$.

- If such a walk exists, then one can make the graph semi-Eulerian by copying the edges traversed several times.
Proposition

The strings having $S$ as a set of $k$-substrings corresponds to walks traversing at least once each edge of the order $k$ de Bruijn graph of $S$.

- If such a walk exists, then one can make the graph semi-Eulerian by copying the edges traversed several times.
- We solve Shortest $S$-Equivalent String by:
Solving shortest equivalent $k$-string via Eulerian walks

Proposition

The strings having $S$ as a set of $k$-substrings corresponds to walks traversing \textit{at least} once each edge of the order $k$ de Bruijn graph of $S$.

- If such a walk exists, then one can make the graph semi-Eulerian by copying the edges traversed several times.
- We solve \textsc{Shortest $S$-Equivalent String} by:
  - Finding a minimal set of edge that can be copied to make the de Bruijn graph semi-Eulerian: we call \textit{Eulerian extension} such a graph with copied edges.
Solving shortest equivalent $k$-string via Eulerian walks

**Proposition**

The strings having $S$ as a set of $k$-substrings corresponds to walks traversing *at least* once each edge of the order $k$ de Bruijn graph of $S$.

- If such a walk exists, then one can make the graph semi-Eulerian by copying the edges traversed several times
- We solve **Shortest $S$-Equivalent String** by:
  - Finding a minimal set of edge that can be copied to make the de Bruijn graph semi-Eulerian: we call *Eulerian extension* such a graph with copied edges
  - Find Eulerian trails in the graph
\[ T = \text{"sing_to_me_of_the_man_muse_the_man_of_twists_and_turns}_
\text{driven_time_and_again_off_course}
\text{once_he_had_plundered}
\text{the_hallowed_heights_of_troy"} \]

\[ k = 3 \]
Flows: definition

To find Eulerian extensions, we use flows:

![Flow Diagram]

1. Add a source node (+1) and a sink node (+1).
2. Connect the source to all nodes with an edge of capacity 1/2.
3. Connect all nodes to the sink with an edge of capacity 1/2.
4. Adjust capacities based on the flow network.

Estéban Gabory (CWI)
Flows: definition

To find Eulerian extensions, we use flows:

![Graph with flow values](image)
Flows: definition

To find Eulerian extensions, we use flows:

\[ c(f) = 3 \cdot 1 + 1 \cdot 2 + 2 \cdot 2 - 1 \cdot 2 + 1 \cdot 2 + 0 \cdot 1 = 12 \]
Construction of a flow problem

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

Now we want to model our extension problem with a flow problem. For this we create bogus nodes and we connect them so the flow can "start" and "end" anywhere on the graph.
Now we want to model our extension problem with a flow problem. We give minimal capacity 1 to each original edge so the flow has to visit each edge at least once.
Construction of a flow problem

Now we want to model our extension problem with a flow problem.
The maximal capacity is a very large integer on each edge, and the costs are 1 on edges from the original graph, 0 on bogus edges.
We can obtain a minimum cost flow in $\tilde{O}(|E|^2 + |E||V|)$ ([Orlin, 1993]).
Link between flows on $G_{\text{ext}}$ and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{O}(|E|^2 + |E||V|)$ ([Orlin, 1993])
- A flow gives us a semi-Eulerian extension of $G$ obtained by copying the edges as many times as they are traversed

\[
\begin{align*}
&\text{We can obtain a minimum cost flow in } \tilde{O}(|E|^2 + |E||V|) \\
&\text{([Orlin, 1993])} \\
&\text{A flow gives us a semi-Eulerian extension of } G \text{ obtained by} \\
&\text{copying the edges as many time as they are traversed}
\end{align*}
\]
We can obtain a minimum cost flow in $\tilde{O}(|E|^2 + |E||V|)$ ([Orlin, 1993])

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 $||W||$ on the extended graph (in $O(||W||)$)
Link between flows on $G_{\text{ext}}$ and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{O}(|E|^2 + |E||V|)$ ([Orlin, 1993])
- 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 $||W||$ on the extended graph (in $O(||W||)$)
- This Eulerian trail corresponds to an Eulerian walk on $G$
Link between flows on $G_{\text{ext}}$ and Eulerian extensions

- We can obtain a minimum cost flow in $\tilde{O}(|E|^2 + |E||V|)$ ([Orlin, 1993])
- 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 $||W||$ on the extended graph (in $O(||W||)$)
- This Eulerian trail corresponds to an Eulerian walk on $G$
- A minimal cost flow gives us a shortest walk
Theorem

The Shortest \textit{S}-Equivalent String problem can be solved in $\tilde{O}(nk + n^2)$ time.
Reduction: The Directed Chinese Postman

**Directed Chinese Postman (DCP)**

**Input:** A directed graph $G(V, E)$.

**Output:** A shortest closed Eulerian walk, or FAIL if that is not possible.
**Theorem**

*Any instance of the Directed Chinese Postman problem can be reduced to an instance of the Shortest \( S \)-Equivalent String problem in linear time.*
Reduction: The Directed Chinese Postman

**Directed Chinese Postman (DCP)**

**Input:** A directed graph $G(V, E)$.

**Output:** A shortest closed Eulerian walk, or FAIL if that is not possible.

**Theorem**

Any instance of the Directed Chinese Postman problem can be reduced to an instance of the Shortest $S$-Equivalent String problem in linear time.

**Theorem**

The Directed Chinese Postman problem can be solved in $\tilde{O}(|E|^2)$ time.
Looking for \( z \) strings

\[ \text{z-Shortest S-Equivalent String: One wants the } z \text{ shortest string instead of only the shortest} \]
Looking for $z$ strings

- **$z$-Shortest $S$-Equivalent String**: One wants the $z$ shortest string instead of only the shortest.

- Finding $z$ minimal cost flows can be done in $\tilde{O}(z|V|^3)$ time, or in $\tilde{O}(z(|E||V| + |V|^2))$ time. ([Könen et al., 2021])
Looking for $z$ strings

- **$z$-Shortest $S$-Equivalent String**: One wants the $z$ shortest string instead of only the shortest.

- Finding $z$ minimal cost flows can be done in $\tilde{O}(z|V|^3)$ time, or in $\tilde{O}(z(|E||V| + |V|^2))$ time. ([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 ([Kurita and Wasa, 2021] or [Conte et al., 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.
Looking for \(z\) strings

- **\(z\text{-Shortest }S\text{-Equivalent String}\):** One wants the \(z\) shortest string instead of only the shortest

- Finding \(z\) minimal cost flows can be done in \(\tilde{O}(z|V|^3)\) time, or in \(\tilde{O}(z(|E||V| + |V|^2))\) time. ([Könen et al., 2021])

- In a semi Eulerian graph, one can compute the set \(W_z\) of the \(z\) smallest Eulerian walks in linear time ([Kurita and Wasa, 2021] or [Conte et al., 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 **\(z\text{-Shortest }S\text{-Equivalent String}\)** problem can be solved in \(\tilde{O}(nk + zn^2 + \|T_z\|)\) time.


A faster strongly polynomial minimum cost flow algorithm. 