Making de Bruijn Graphs Eulerian

Giulia Bernardini\textsuperscript{1,2}, Huiping Chen\textsuperscript{3}, Grigorios Loukides\textsuperscript{3}, Solon P. Pissis\textsuperscript{2,4}, Leen Stougie\textsuperscript{2,4} and Michelle Sweering\textsuperscript{2}

\textsuperscript{1}University of Trieste, Trieste, \textsuperscript{2}CWI, Amsterdam \textsuperscript{3}King’s College London \textsuperscript{4}Vrije Universiteit, Amsterdam

27\textsuperscript{th} June 2022
Outline

Definitions
- Graphs
- Strings

Problem: Eulerian Extension

Hardness

Algorithm

Results
A graph $G$ consists of
- a finite set of nodes $V$
- a finite set of edges $E$
  connecting the nodes
A **graph** $G$ consists of

- a finite set of nodes $V$
- a finite set of edges $E$ connecting the nodes

In a **directed graph** the edges $E \subseteq \{(u, v) : u, v \in V\}$ have a direction associated with them.
A graph $G$ consists of
- a finite set of nodes $V$
- a finite set of edges $E$
  connecting the nodes

In a directed graph the edges $E \subseteq \{(u, v) : u, v \in V\}$ have a direction associated with them.

In a multigraph we can have multiple copies of each edge.
Graphs

A graph $G$ consists of

- a finite set of nodes $V$
- a finite set of edges $E$ connecting the nodes

In a directed graph the edges $E \subseteq \{(u, v) : u, v \in V\}$ have a direction associated with them.

In a multigraph we can have multiple copies of each edge.
Graphs

A graph $G$ consists of

- a finite set of nodes $V$
- a finite set of edges $E$ connecting the nodes

In a directed graph the edges $E \subseteq \{(u, v) : u, v \in V\}$ have a direction associated with them.

In a multigraph we can have multiple copies of each edge.

Our problem

We work with directed multigraphs.
More Graph Definitions

**Walk**
Sequence of nodes connected by edges

**Circuit**
Walk with the same first and last node
Eulerian walk/circuit

A walk/circuit which visits every edge exactly once.
Euler’s Theorem

Theorem

A graph contains a Eulerian circuit if and only if

- the edges are connected and
- the nodes are balanced.
De Bruijn Graphs

Complete order 3 de Bruijn graph over alphabet $\Sigma = \{a, b\}$
De Bruijn Graphs

Complete order 3 de Bruijn graph over alphabet \( \Sigma = \{a, b\} \)

Order 3 de Bruijn graph of \( abba \)
Problem: Eulerian Extension

Outline

Definitions

Problem: Eulerian Extension

Hardness

Algorithm

Results
Problem (Eulerian Extension)

We are given a multigraph $G = (V, E)$ where $V \subseteq \mathcal{V}$ and a set of forbidden edges $F \subseteq \mathcal{V} \times \mathcal{V}$. Find a minimum multiset of feasible edges $A \subseteq (\mathcal{V} \times \mathcal{V}) \setminus F$ and a set of nodes $B \subseteq \mathcal{V}$ such that

- $(V \cup B, E \cup A)$ is connected and
- $(V \cup B, E \cup A)$ is balanced.
Our Setting

\[ G = (V, E) \] is a de Bruijn graph of strings
Our Setting

$G = (V, E)$ is a de Bruijn graph of strings

**Extend-DBG** $V = \Sigma^{k-1}$ and $F$ is all edges not in the complete de Bruijn graph.
Our Setting

\( G = (V, E) \) is a de Bruijn graph of strings

**Extend-DBG** \( \mathcal{V} = \Sigma^{k-1} \) and \( F \) is all edges not in the complete de Bruijn graph.

**R-Extend-DBG** \( \mathcal{V} = V \) and \( F \) is all edges not in the complete de Bruijn graph.
Outline

Definitions

Problem: Eulerian Extension

Hardness

Algorithm

Results
**Theorem**

*Restricted Eulerian Extension is NP-hard (even if the graph is a de Bruijn graph).*
Outline

Definitions

Problem: Eulerian Extension

Hardness

Algorithm

Results
Connect and Balance (CAB)

Idea

We do not solve the Eulerian Extension problem optimally.

Bernardini et al.
Making de Bruijn Graphs Eulerian
CPM 2022 14 / 20
Connect and Balance (CAB)

Idea

1. Connect the graph

Remark

We do not solve the Eulerian Extension problem optimally.

Bernardini et al.  Making de Bruijn Graphs Eulerian  CPM 2022
Connect and Balance (CAB)

Idea

1. Connect the graph
2. Balance the graph
Connect and Balance (CAB)

Idea

1. Connect the graph
2. Balance the graph

Remark
We do not solve the Eulerian Extension problem optimally.
Connecting

Kruskal’s algorithm:

- Connect closest components
- Optimal for minimum spanning tree
Connecting

**Idea 1**

Use graph algorithms.

- Complete de Bruijn graph has $|\Sigma|^{k-1}$ nodes
Connecting

Idea 1
Use graph algorithms.

- Complete de Bruijn graph has $|\Sigma|^{k-1}$ nodes

Idea 2
Use string algorithms to find the minimum distance between all pairs of nodes in $G = (V, E)$.

- $O(k|V|^2)$
Idea 1
Use graph algorithms.

- Complete de Bruijn graph has $|\Sigma|^{k-1}$ nodes

Idea 2
Use string algorithms to find the minimum distance between all pairs of nodes in $G = (V, E)$.

- $O(k|V|^2)$

Idea 3
Use an automaton to simultaneously compute all overlaps and then go through them from longest to shortest.

$+ O(k|V| \log |V| + |E|)$
Aho-Corasick Automaton
Balancing

\[ d^+(v) = \text{number of outgoing edges} \]

\[ d^-(v) = \text{number of incoming edges} \]
Balancing

\[ d^+(v) = \text{number of outgoing edges} \]
\[ d^-(v) = \text{number of incoming edges} \]

Two types of unbalanced nodes:

- \( Z^+ = \{v \mid d^+(v) > d^-(v)\} \)
- \( Z^- = \{v \mid d^-(v) > d^+(v)\} \)
Balancing

\[ d^+(v) = \text{number of outgoing edges} \]
\[ d^-(v) = \text{number of incoming edges} \]

Two types of unbalanced nodes:

- \( Z^+ = \{ v \mid d^+(v) > d^-(v) \} \)
- \( Z^- = \{ v \mid d^-(v) > d^+(v) \} \)

Idea

Use a similar automaton with only links from \( Z^- \) to \( Z^+ \).
Balancing

\[ d^+(v) = \text{number of outgoing edges} \]
\[ d^-(v) = \text{number of incoming edges} \]

Two types of unbalanced nodes:
- \[ Z^+ = \{ v \mid d^+(v) > d^-(v) \} \]
- \[ Z^- = \{ v \mid d^-(v) > d^+(v) \} \]

Idea
Use a similar automaton with only links from \( Z^- \) to \( Z^+ \).

Time Complexity:
\[ O(|V|k + |E|) \]
Outline

Definitions

Problem: Eulerian Extension

Hardness

Algorithm

Results
Results

Algorithm: CAB (connect and balance)
Results

Algorithm: CAB (connect and balance)

Benchmarks:
Results

Algorithm: CAB (connect and balance)

Benchmarks:
- MGR (multi-SCS greedy)
Results

Algorithm: CAB (connect and balance)

Benchmarks:
- MGR (multi-SCS greedy)
- SAB (SCS and balance)
Results

Algorithm: CAB (connect and balance)

Benchmarks:
- MGR (multi-SCS greedy)
- SAB (SCS and balance)