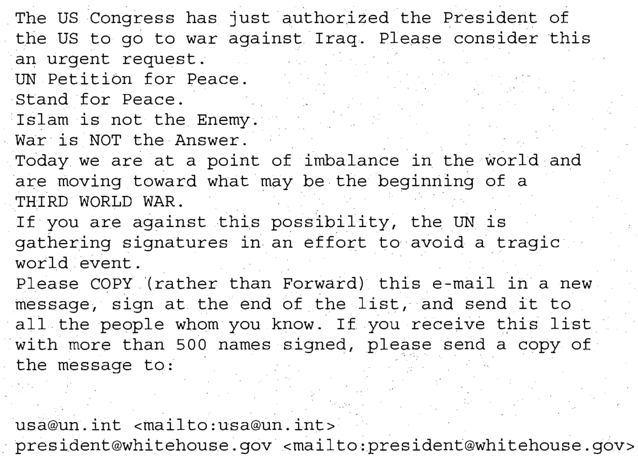
Summarizing Diverging String Sequences, with Applications to Chain-Letter Petitions

Patty Commins[1,2], David Liben-Nowell[1],
Tina Liu[1,3], and Kiran Tomlinson[1,4]

[1]Department of Computer Science, Carleton College
[2]Department of Mathematics, University of Minnesota
[3]Surescripts
[4]Department of Computer Science, Cornell University

CPM 2020

Chain-Letter Petitions



```
The US Congress has just authorized the President of
the US to go to war against Iraq. Please consider this
an urgent request.
UN Petition for Peace.
Stand for Peace.
Islam is not the Enemy.
War is NOT the Answer.
Today we are at a point of imbalance in the world and
are moving toward what may be the beginning of a
THIRD WORLD WAR.
If you are against this possibility, the UN is
gathering signatures in an effort to avoid a tragic
world event.
Please COPY (rather than Forward) this e-mail in a new
message, sign at the end of the list, and send it to
all the people whom you know. If you receive this list
with more than 500 names signed, please send a copy of
the message to:


usa@un.int <mailto:usa@un.int>
president@whitehouse.gov <mailto:president@whitehouse.gov>
```

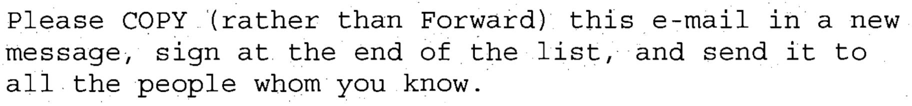
$\sim$ 3.5m emails

$\sim$ 170k signers

(Chierichetti, Kleinberg, & Liben-Nowell 2011)

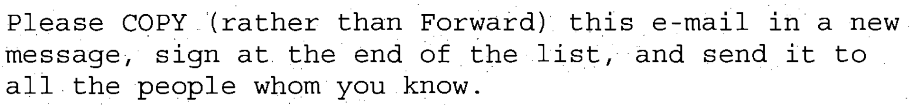
Sent 20 February 2003, retrieved from G.W.B. Presidential Library

Chain-Letter Petitions



Please COPY (rather than Forward) this e-mail in a new message, sign at the end of the list, and send it to all the people whom you know.

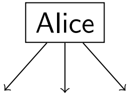
Chain-Letter Petitions


Please COPY (rather than Forward) this e-mail in a new message, sign at the end of the list, and send it to all the people whom you know.

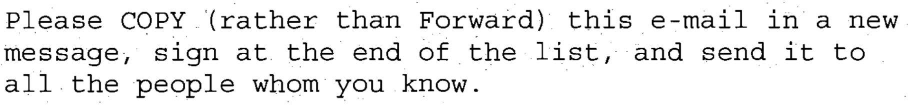
Alice

Chain-Letter Petitions

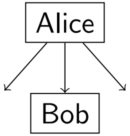
Please COPY (rather than Forward) this e-mail in a new message, sign at the end of the list, and send it to all the people whom you know.



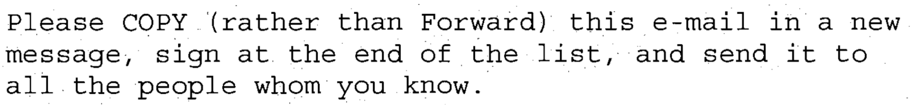
Chain-Letter Petitions



Please COPY (rather than Forward) this e-mail in a new message, sign at the end of the list, and send it to all the people whom you know.



Chain-Letter Petitions



Please COPY (rather than Forward) this e-mail in a new message, sign at the end of the list, and send it to all the people whom you know.

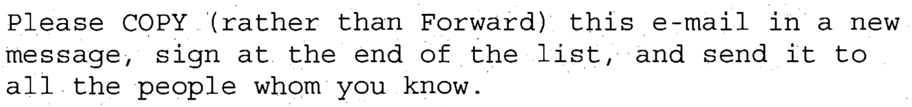
$$\begin{tikzcd}
 & \boxed{\text{Alice}} \arrow[d] & \\
 & \boxed{\text{Bob}} \arrow[dl] \arrow[dr] & \\
 {} & & {}
\end{tikzcd}$$

Chain-Letter Petitions

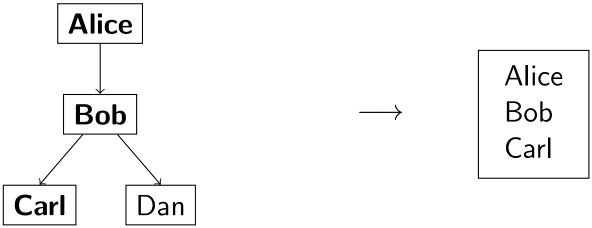
Please COPY (rather than Forward) this e-mail in a new message, sign at the end of the list, and send it to all the people whom you know.

$$\begin{tikzcd}
 & \boxed{\text{Alice}} \arrow[d] & \\
 & \boxed{\text{Bob}} \arrow[dl] \arrow[dr] & \\
\boxed{\text{Carl}} & & \boxed{\text{Dan}}
\end{tikzcd}$$

Chain-Letter Petitions

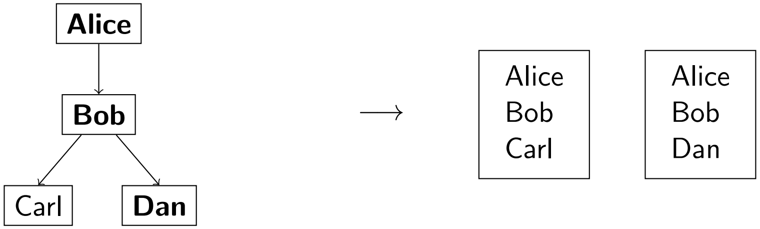


Please COPY (rather than Forward) this e-mail in a new message, sign at the end of the list, and send it to all the people whom you know.

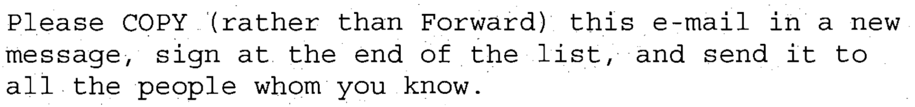


Chain-Letter Petitions

```
Please COPY (rather than Forward) this e-mail in a new
message, sign at the end of the list, and send it to
all the people whom you know.
```



Chain-Letter Petitions



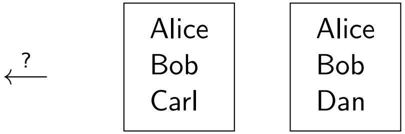
Please COPY (rather than Forward) this e-mail in a new message, sign at the end of the list, and send it to all the people whom you know.

Alice	Alice
Bob	Bob
Carl	Dan

Reconstruction

Central Question

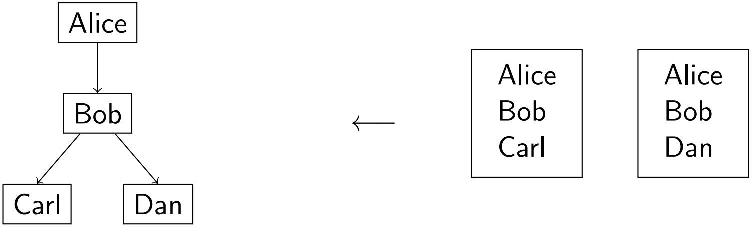
Can we reconstruct the propagation tree from signature lists?



Reconstruction

Central Question

Can we reconstruct the propagation tree from signature lists?



Challenge: Mutations

People are bad at copy-paste.

Challenge: Mutations

People are bad at copy-paste.

1. Substitution

$$\boxed{\begin{array}{l}\text{Alice}\\ \textbf{Bob}\\ \text{Carl}\end{array}} \longrightarrow \boxed{\begin{array}{l}\text{Alice}\\ \textbf{Eve}\\ \text{Carl}\end{array}}$$

Challenge: Mutations

People are bad at copy-paste.

1. Substitution

$$\boxed{\begin{array}{l}\text{Alice}\\ \textbf{Bob}\\ \text{Carl}\end{array}} \longrightarrow \boxed{\begin{array}{l}\text{Alice}\\ \textbf{Eve}\\ \text{Carl}\end{array}}$$

2. Insertion

$$\boxed{\begin{array}{l}\text{Alice}\\ \text{Bob}\\ \text{Carl}\end{array}} \longrightarrow \boxed{\begin{array}{l}\text{Alice}\\ \text{Bob}\\ \textbf{Eve}\\ \text{Carl}\end{array}}$$

Challenge: Mutations

People are bad at copy-paste.

1. Substitution

Alice	$\longrightarrow$	Alice
Bob		**Eve**
Carl		Carl

2. Insertion

Alice	$\longrightarrow$	Alice
Bob		Bob
Carl		**Eve**
		Carl

3. Deletion

Alice	$\longrightarrow$	Alice
Bob		Carl
Carl		

Challenge: Mutations

People are bad at copy-paste.

1. Substitution

Alice **Bob** Carl	$\longrightarrow$	Alice **Eve** Carl

2. Insertion

Alice Bob Carl	$\longrightarrow$	Alice Bob **Eve** Carl

Character-level:
Carl $\rightarrow$ Car**ol**, Al**i**ce $\rightarrow$ Al**y**ce

3. Deletion

Alice **Bob** Carl	$\longrightarrow$	Alice Carl

Challenge: Mutations

People are bad at copy-paste.

1. Substitution

Alice **Bob** Carl	$\longrightarrow$	Alice **Eve** Carl

2. Insertion

Alice Bob Carl	$\longrightarrow$	Alice Bob **Eve** Carl

3. Deletion

Alice **Bob** Carl	$\longrightarrow$	Alice Carl

Character-level:
Carl $\rightarrow$ Car**ol**, Al**i**ce $\rightarrow$ Al**y**ce

All present in the Iraq War petition
(Liben-Nowell & Kleinberg 2008)

Reconstruction with Mutations

$$\boxed{\begin{array}{l}\text{Alice}\\\text{Carl}\\\text{Eve}\end{array}} \quad \boxed{\begin{array}{l}\text{Alice}\\\text{Bob}\\\text{Carol}\\\text{Dan}\end{array}} \quad \boxed{\begin{array}{l}\text{Alice}\\\text{Bob}\\\text{Carl}\\\text{Frank}\end{array}} \qquad \xrightarrow{?}$$

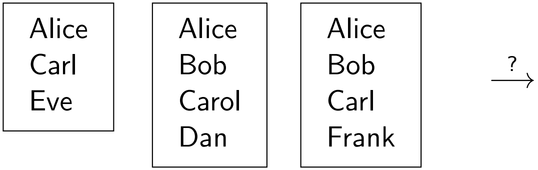
Reconstruction with Mutations

Alice	Alice	Alice
Carl	Bob	Bob
Eve	Carol	Carl
	Dan	Frank

$\xrightarrow{?}$

Key chain letter features

Reconstruction with Mutations



Key chain letter features

1. One-ended growth

Reconstruction with Mutations

Alice	Alice	Alice
Carl	Bob	Bob
Eve	Carol	Carl
	Dan	Frank

$\xrightarrow{?}$

Key chain letter features

1. One-ended growth
2. Divergence

Reconstruction with Mutations

Alice
Carl
Eve

Alice
Bob
Carol
Dan

Alice
Bob
Carl
Frank

$\xrightarrow{?}$

Key chain letter features

1. One-ended growth
2. Divergence
3. Mutation with inheritance

Summary of Contributions

1. Formal definition of chain letter reconstruction problem

Summary of Contributions

1. Formal definition of chain letter reconstruction problem
2. NP-hardness proof

Summary of Contributions

1. Formal definition of chain letter reconstruction problem
2. NP-hardness proof
3. Efficient optimal solution for two lists

Summary of Contributions

1. Formal definition of chain letter reconstruction problem
2. NP-hardness proof
3. Efficient optimal solution for two lists
4. Fixed-parameter tractable: poly-time algorithm for $O(1)$ lists

Summary of Contributions

1. Formal definition of chain letter reconstruction problem
2. NP-hardness proof
3. Efficient optimal solution for two lists
4. Fixed-parameter tractable: poly-time algorithm for $O(1)$ lists
5. Fast heuristic for arbitrary number of lists

Summary of Contributions

1. Formal definition of chain letter reconstruction problem
2. NP-hardness proof
3. Efficient optimal solution for two lists
4. Fixed-parameter tractable: poly-time algorithm for $O(1)$ lists
5. Fast heuristic for arbitrary number of lists
6. Experimental evaluation on synthetic data

Summary of Contributions

1. Formal definition of chain letter reconstruction problem
2. NP-hardness proof*
3. Efficient optimal solution for two lists
4. Fixed-parameter tractable: poly-time algorithm for $O(1)$ lists*
5. Fast heuristic for arbitrary number of lists
6. Experimental evaluation on synthetic data

* see paper

Related Work

- Chain letters
    - Iraq war petition tree structure (Liben-Nowell & Kleinberg 2008; Golub & Jackson 2010; Chierichetti, Liben-Nowell, & Kleinberg 2011)
    - Tree reconstruction from plea (Bennett, Li, & Ma 2003)

Related Work

- Chain letters
  - Iraq war petition tree structure (Liben-Nowell & Kleinberg 2008; Golub & Jackson 2010; Chierichetti, Liben-Nowell, & Kleinberg 2011)
  - Tree reconstruction from plea (Bennett, Li, & Ma 2003)
- One-ended growth and divergence
  - Trie (De La Briandais 1959; Fredkin 1960)
  - Online conversations (Kumar, Mahdian, & McGlohon 2010)

Related Work

- Chain letters
    - Iraq war petition tree structure (Liben-Nowell & Kleinberg 2008; Golub & Jackson 2010; Chierichetti, Liben-Nowell, & Kleinberg 2011)
    - Tree reconstruction from plea (Bennett, Li, & Ma 2003)
- One-ended growth and divergence
    - Trie (De La Briandais 1959; Fredkin 1960)
    - Online conversations (Kumar, Mahdian, & McGlohon 2010)
- Divergence and mutation
    - Molecular phylogenetics (Yang & Rannala 2012)
    - Stories; e.g., Little Red Riding Hood (Tehrani 2013)

Outline

Problem Definition, Informally

DSSSP (Diverging String Sequence Summarization Problem)

Given diverging string sequences:

Alice	Alice	Alice
Carl	Bob	Bob
Eve	Carol	Carl
	Dan	Frank

Problem Definition, Informally

DSSSP (Diverging String Sequence Summarization Problem)

Given **diverging string sequences**:

Alice	**Alice**	Alice
Carl	**Bob**	Bob
Eve	**Carol**	Carl
	Dan	Frank

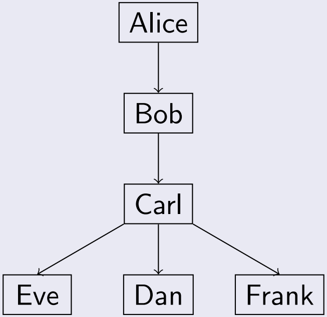
Problem Definition, Informally

DSSSP (Diverging String Sequence Summarization Problem)

Given diverging string sequences:

Alice Carl Eve	Alice Bob Carol Dan	Alice Bob Carl Frank

Find best summary tree:


$$\begin{tikzcd}
 & \boxed{\text{Alice}} \arrow[d] & \\
 & \boxed{\text{Bob}} \arrow[d] & \\
 & \boxed{\text{Carl}} \arrow[dl] \arrow[d] \arrow[dr] & \\
\boxed{\text{Eve}} & \boxed{\text{Dan}} & \boxed{\text{Frank}}
\end{tikzcd}$$

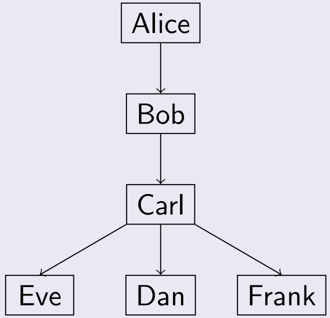
Problem Definition, Informally

DSSSP (Diverging String Sequence Summarization Problem)

Given diverging string sequences:

Alice	Alice	Alice
Carl	Bob	Bob
Eve	Carol	Carl
	Dan	Frank

Find **best** summary tree:



Competing Objectives

Alice
Carl
Eve

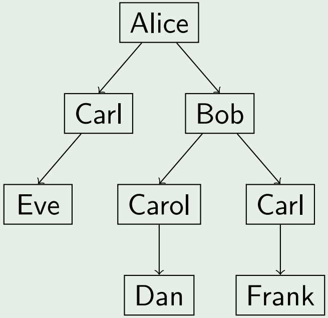
Alice
Bob
Carol
Dan

Alice
Bob
Carl
Frank

Competing Objectives

Alice	Alice	Alice
Carl	Bob	Bob
Eve	Carol	Carl
	Dan	Frank

Accurate representation



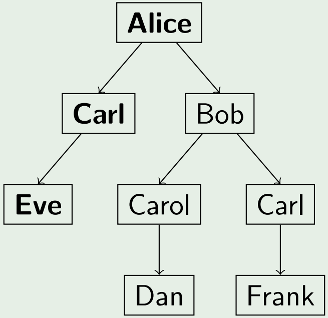
Competing Objectives

Alice
Carl
Eve

Alice
Bob
Carol
Dan

Alice
Bob
Carl
Frank

Accurate representation



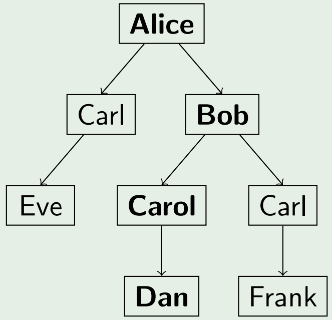
Competing Objectives

Alice
Carl
Eve

Alice
Bob
Carol
Dan

Alice
Bob
Carl
Frank

Accurate representation



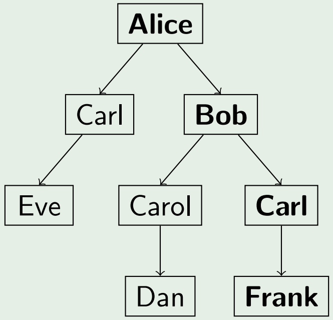
Competing Objectives

Alice
Carl
Eve

Alice
Bob
Carol
Dan

Alice
Bob
Carl
Frank

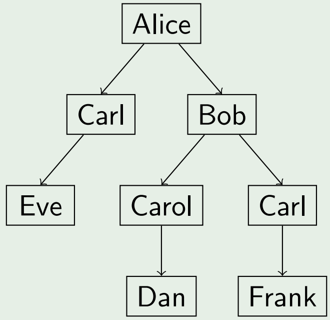
Accurate representation



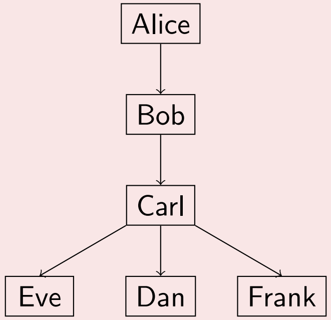
Competing Objectives

Alice	Alice	Alice
Carl	Bob	Bob
Eve	Carol	Carl
	Dan	Frank

Accurate representation



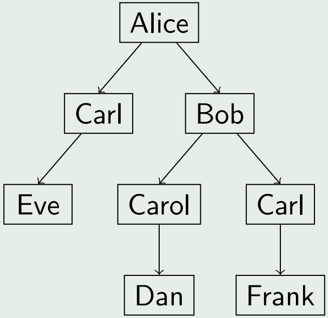
Minimal redundancy



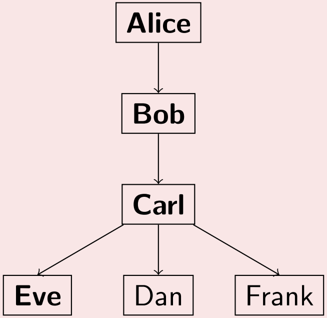
Competing Objectives

Alice	Alice	Alice
Carl	Bob	Bob
Eve	Carol	Carl
	Dan	Frank

Accurate representation



Minimal redundancy



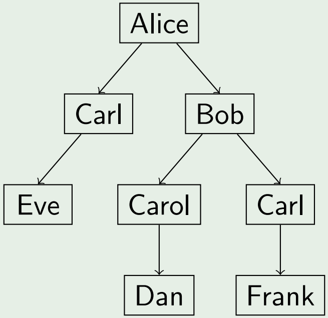
Competing Objectives

Alice
Carl
Eve

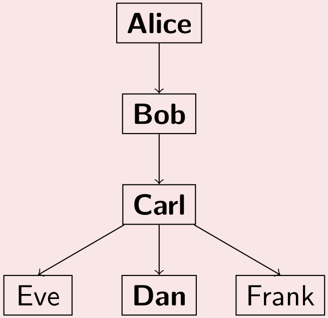
Alice
Bob
Carol
Dan

Alice
Bob
Carl
Frank

Accurate representation



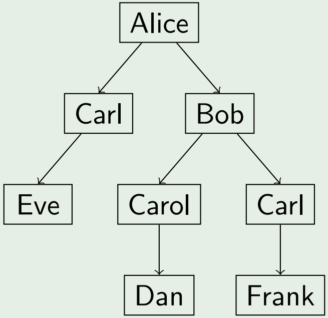
Minimal redundancy



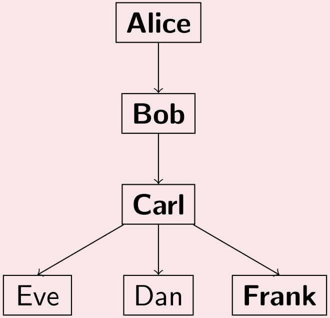
Competing Objectives

Alice Carl Eve	Alice Bob Carol Dan	**Alice** **Bob** **Carl** **Frank**

Accurate representation

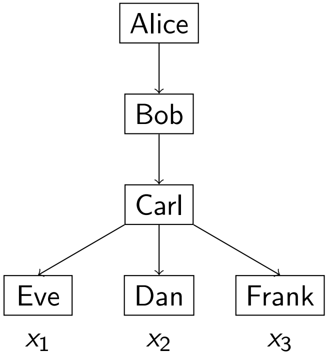


Minimal redundancy



Measuring Representation Accuracy

x_1	x_2	x_3
Alice	Alice	Alice
Carl	Bob	Bob
Eve	Carol	Carl
	Dan	Frank



Measuring Representation Accuracy

x_1	x_2	x_3
Alice	Alice	Alice
Carl	Bob	Bob
Eve	Carol	Carl
	Dan	Frank

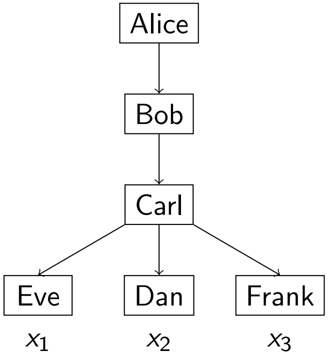
$\text{labelseq}_T(x_1)$	$\text{labelseq}_T(x_2)$	$\text{labelseq}_T(x_3)$
Alice	Alice	Alice
Bob	Bob	Bob
Carl	Carl	Carl
Eve	Dan	Frank

$$\begin{tikzcd}
& \boxed{\text{Alice}} \arrow[d] & \\
& \boxed{\text{Bob}} \arrow[d] & \\
& \boxed{\text{Carl}} \arrow[dl] \arrow[d] \arrow[dr] & \\
\boxed{\text{Eve}} & \boxed{\text{Dan}} & \boxed{\text{Frank}} \\
x_1 & x_2 & x_3
\end{tikzcd}$$

Measuring Representation Accuracy

x_1	x_2	x_3
Alice Carl Eve	Alice Bob Carol Dan	Alice Bob Carl Frank

labelseq$_T(x_1)$	labelseq$_T(x_2)$	labelseq$_T(x_3)$
Alice Bob Carl Eve	Alice Bob Carl Dan	Alice Bob Carl Frank



AED(x, y)

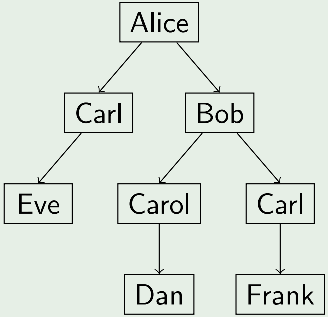
Allowed operations:

1. Insert string into x
2. Substitute string

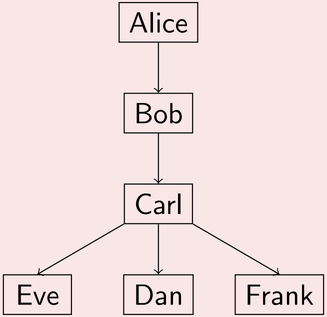
Costs using Levenshtein ED

Minimizing Redundancy

Accurate representation

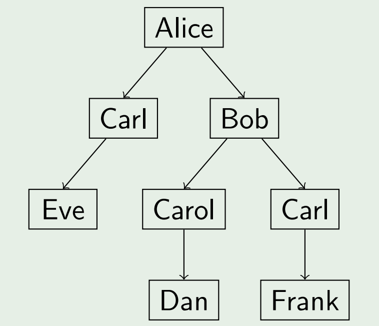


Minimal redundancy

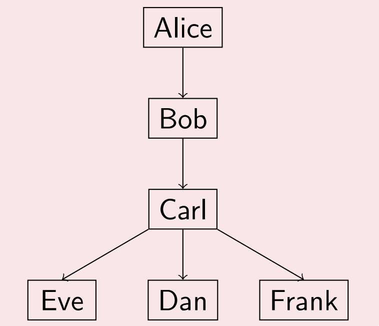


Minimizing Redundancy

Accurate representation



Minimal redundancy

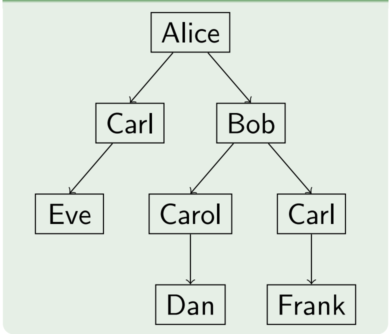


8 nodes

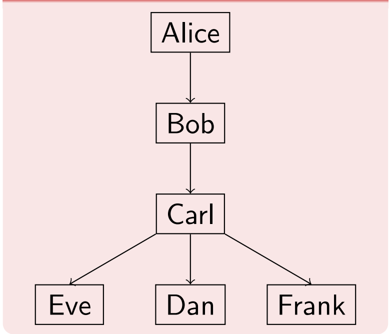
6 nodes

Minimizing Redundancy

Accurate representation



Minimal redundancy



8 nodes

6 nodes

$\Rightarrow$ Cost λ per node

Problem Definition, Formally

DSSSP

Given diverging string sequences $x_1, \ldots, x_m$ and node cost λ, find tree T that minimizes

$$\mathrm{err}_\lambda(T) = \underbrace{\sum_{i=1}^{m} \mathrm{AED}(x_i, \mathrm{labelseq}_T(x_i))}_{\text{loss}} + \underbrace{\lambda \cdot |T|}_{\text{regularization}}$$

Outline

Two sequences: dynamic programming

Two string sequences x, y; align $x_{i\dots}$ and $y_{j\dots}$

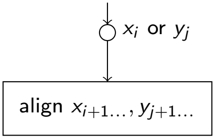
$$\mathrm{EDG}(i,j) = \min \left\{ \vphantom{\begin{matrix}a\\b\\c\\d\end{matrix}} \right.$$

Two sequences: dynamic programming

Two string sequences x, y; align $x_{i\ldots}$ and $y_{j\ldots}$

$$\mathrm{EDG}(i,j) = \min \begin{cases} \mathrm{EDG}(i+1, j+1) + \lambda + \mathrm{ED}(x_i, y_j) & \text{(substitution)} \\ \\ \\ \end{cases}$$

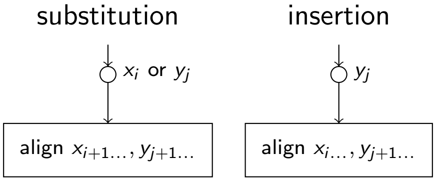
substitution



Two sequences: dynamic programming

Two string sequences x, y; align $x_{i\ldots}$ and $y_{j\ldots}$

$$\mathrm{EDG}(i,j) = \min \begin{cases} \mathrm{EDG}(i+1, j+1) + \lambda + \mathrm{ED}(x_i, y_j) & \text{(substitution)} \\ \mathrm{EDG}(i, j+1) \quad\quad + \lambda + \mathrm{ED}(\varepsilon, y_j) & \text{(insertion)} \\ & \end{cases}$$

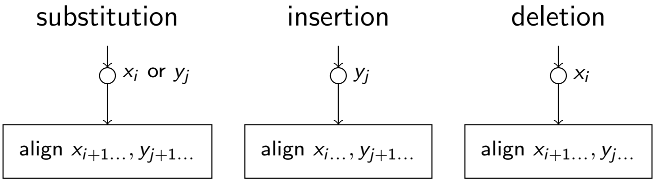


substitution
x_i or y_j
align $x_{i+1\ldots}, y_{j+1\ldots}$
insertion
y_j
align $x_{i\ldots}, y_{j+1\ldots}$

Two sequences: dynamic programming

Two string sequences x, y; align $x_{i\dots}$ and $y_{j\dots}$

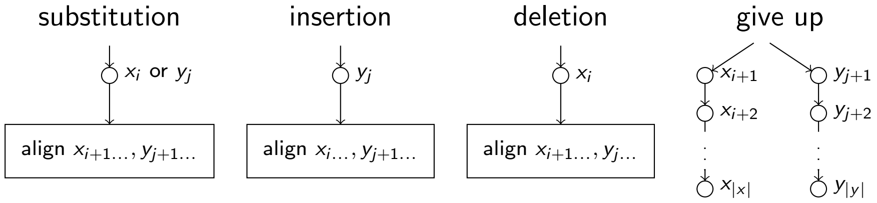
$$\mathrm{EDG}(i,j) = \min \begin{cases} \mathrm{EDG}(i+1, j+1) + \lambda + \mathrm{ED}(x_i, y_j) & \text{(substitution)} \\ \mathrm{EDG}(i, j+1) + \lambda + \mathrm{ED}(\varepsilon, y_j) & \text{(insertion)} \\ \mathrm{EDG}(i+1, j) + \lambda + \mathrm{ED}(x_i, \varepsilon) & \text{(deletion)} \end{cases}$$


substitution
x_i or y_j
align $x_{i+1\dots}, y_{j+1\dots}$
insertion
y_j
align $x_{i\dots}, y_{j+1\dots}$
deletion
x_i
align $x_{i+1\dots}, y_{j\dots}$

Two sequences: dynamic programming

Two string sequences x, y; align $x_{i\ldots}$ and $y_{j\ldots}$

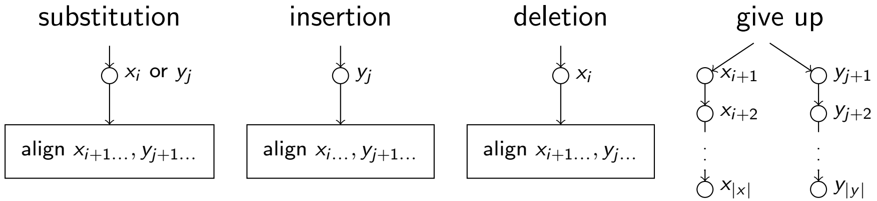
$$\mathrm{EDG}(i,j) = \min \begin{cases} \mathrm{EDG}(i+1, j+1) + \lambda + \mathrm{ED}(x_i, y_j) & \text{(substitution)} \\ \mathrm{EDG}(i, j+1) + \lambda + \mathrm{ED}(\varepsilon, y_j) & \text{(insertion)} \\ \mathrm{EDG}(i+1, j) + \lambda + \mathrm{ED}(x_i, \varepsilon) & \text{(deletion)} \\ \lambda(|x| - i + 1) + \lambda(|y| - j + 1) & \text{(give up)} \end{cases}$$



Two sequences: dynamic programming

Two string sequences x, y; align $x_{i\ldots}$ and $y_{j\ldots}$

$$\mathrm{EDG}(i,j) = \min \begin{cases} \mathrm{EDG}(i+1, j+1) + \lambda + \mathrm{ED}(x_i, y_j) & \text{(substitution)} \\ \mathrm{EDG}(i, j+1) + \lambda + \mathrm{ED}(\varepsilon, y_j) & \text{(insertion)} \\ \mathrm{EDG}(i+1, j) + \lambda + \mathrm{ED}(x_i, \varepsilon) & \text{(deletion)} \\ \lambda(|x| - i + 1) + \lambda(|y| - j + 1) & \text{(give up)} \end{cases}$$



Theorem

This produces an optimal two-sequence DSSSP solution.

Algorithm for more sequences

Theorem

DSSSP is NP-hard with an unbounded number of sequences.

Algorithm for more sequences

Theorem

DSSSP is NP-hard with an unbounded number of sequences.

Idea: progressive alignment (Feng & Doolittle 1987)

Repeatedly merge pair of sequences that diverges last

Algorithm for more sequences

Theorem

DSSSP is NP-hard with an unbounded number of sequences.

Idea: progressive alignment (Feng & Doolittle 1987)

Repeatedly merge pair of sequences that diverges last

x_1	x_2	x_3
Alice	Alice	Alice
Carl	Bob	Bob
Eve	Carol	Carl
	Dan	Frank

Algorithm for more sequences

Theorem

DSSSP is NP-hard with an unbounded number of sequences.

Idea: progressive alignment (Feng & Doolittle 1987)

Repeatedly merge pair of sequences that diverges last

x_1	x_2	x_3
Alice	Alice	Alice
Carl	Bob	Bob
Eve	Carol	Carl
	Dan	Frank

$$\xrightarrow{\text{All pairwise EDG alignments}}$$

x_1		x_2
Alice	$=$	Alice
	$<$	Bob
Carl	$=$	Carol
Eve		Dan

x_1		x_3
Alice	$=$	Alice
	$<$	Bob
Carl	$=$	Carl
Eve		Frank

x_2		x_3
Alice	$=$	Alice
Bob	$=$	Bob
Carol	$=$	Carl
Dan		Frank

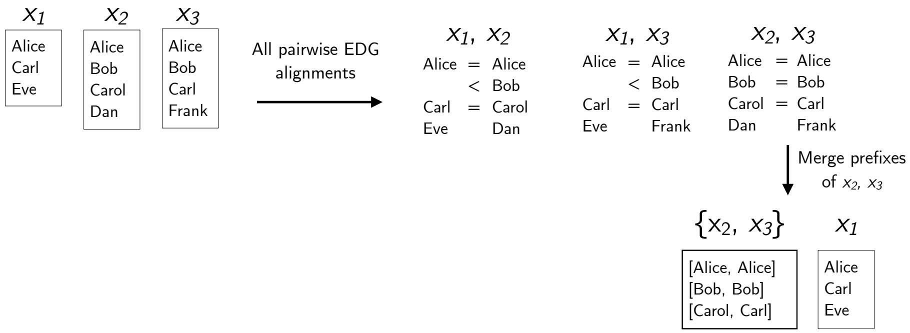
Algorithm for more sequences

Theorem

DSSSP is NP-hard with an unbounded number of sequences.

Idea: progressive alignment (Feng & Doolittle 1987)

Repeatedly merge pair of sequences that diverges last



x_1	x_2	x_3
Alice	Alice	Alice
Carl	Bob	Bob
Eve	Carol	Carl
	Dan	Frank

x_1, x_2

Alice	=	Alice
	<	Bob
Carl	=	Carol
Eve		Dan

x_1, x_3

Alice	=	Alice
	<	Bob
Carl	=	Carl
Eve		Frank

x_2, x_3

Alice	=	Alice
Bob	=	Bob
Carol	=	Carl
Dan		Frank

$\{x_2, x_3\}$	x_1
[Alice, Alice]	Alice
[Bob, Bob]	Carl
[Carol, Carl]	Eve

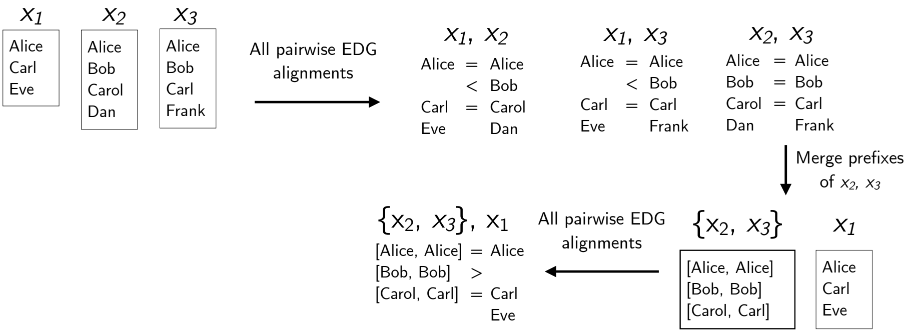
Algorithm for more sequences

Theorem

DSSSP is NP-hard with an unbounded number of sequences.

Idea: progressive alignment (Feng & Doolittle 1987)

Repeatedly merge pair of sequences that diverges last



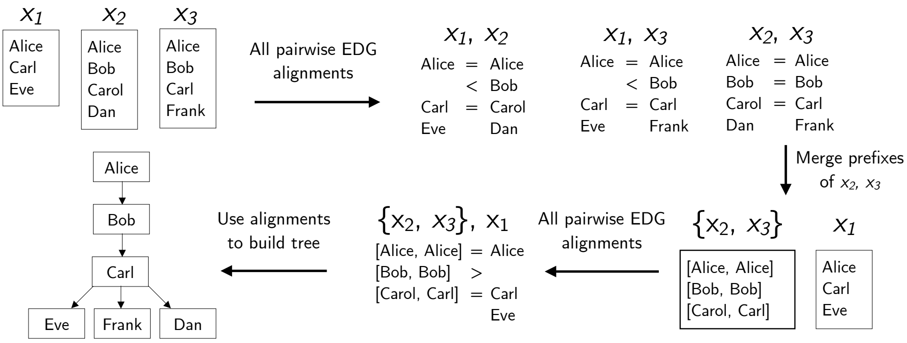
Algorithm for more sequences

Theorem

DSSSP is NP-hard with an unbounded number of sequences.

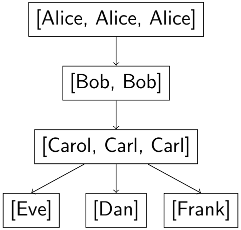
Idea: progressive alignment (Feng & Doolittle 1987)

Repeatedly merge pair of sequences that diverges last



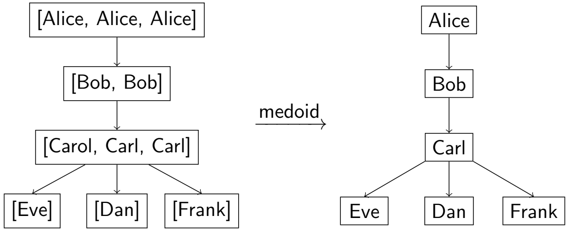
Algorithm details

1. Labeling the final tree



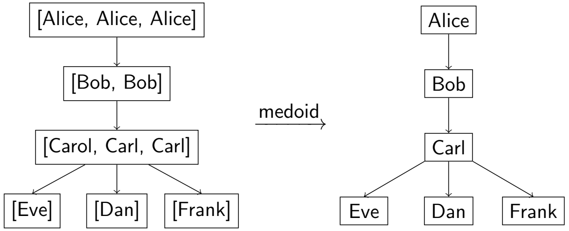
Algorithm details

1. Labeling the final tree



Algorithm details

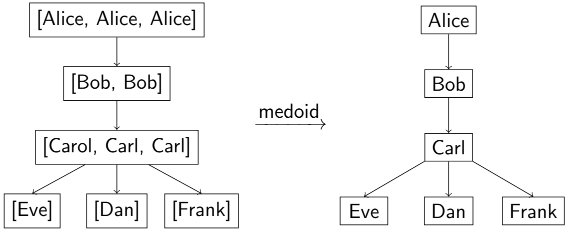
1. Labeling the final tree



median NP-hard
(de la Higuera & Casacuberta 2000)

Algorithm details

1. Labeling the final tree

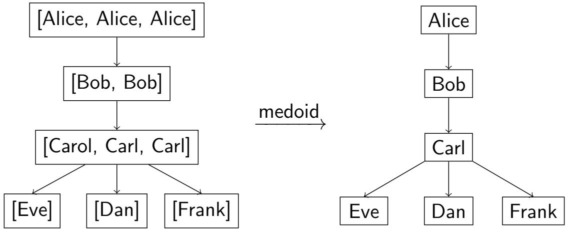


median NP-hard
(de la Higuera & Casacuberta 2000)

2. Generalizing EDG to sequences of *lists* of strings

Algorithm details

1. Labeling the final tree



median NP-hard
(de la Higuera & Casacuberta 2000)

2. Generalizing EDG to sequences of *lists* of strings

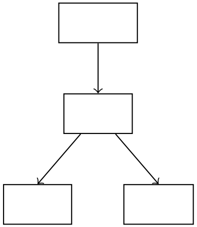
   Substitution cost for lists A, B:

$$\mathcal{C}(A, B) := (\text{AED error if we merge } A, B) - (\text{AED error if we don't})$$

Outline

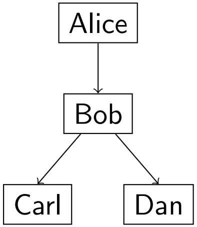
Generating synthetic data

1. Run branching process (Watson & Galton 1875)



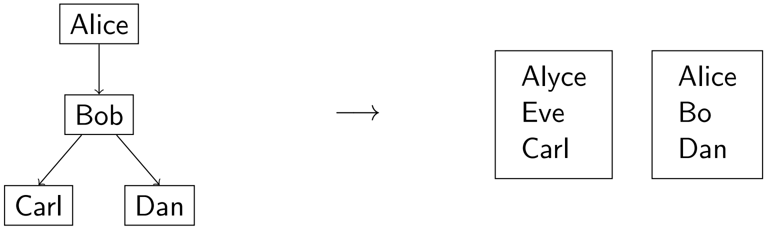
Generating synthetic data

1. Run branching process (Watson & Galton 1875)
2. Label with random strings



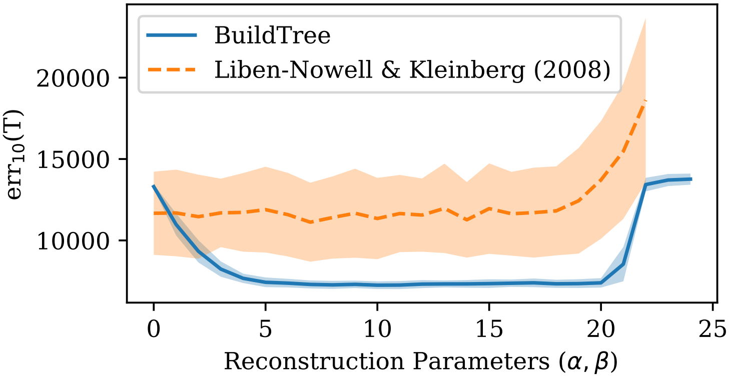
Generating synthetic data

1. Run branching process (Watson & Galton 1875)
2. Label with random strings
3. Simulate noisy propagation down the tree



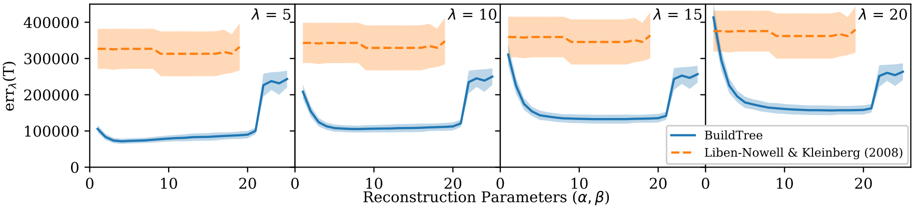
Good performance across a range of node costs

15 sequences, 500 trials



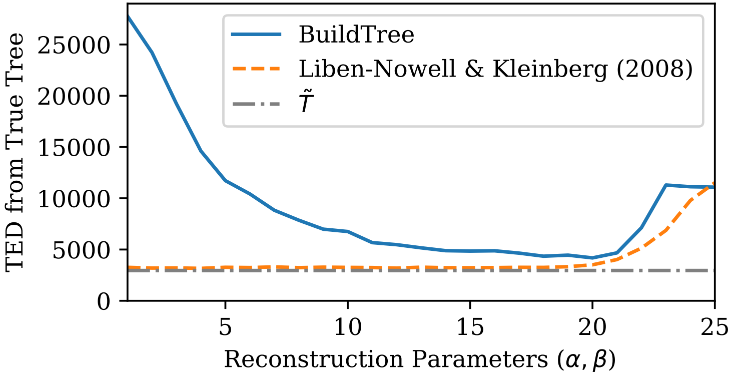
Larger performance gap with more sequences

100 sequences, 8 trials



Approximate comparison with true tree

15 sequences, 500 trials



Outline

Takeaways and open questions

Takeaways

1. Chain letter petitions exhibit one-ended growth, divergence, and mutation: intriguing reconstruction problem

Takeaways and open questions

Takeaways

1. Chain letter petitions exhibit one-ended growth, divergence, and mutation: intriguing reconstruction problem
2. NP-hard in general, but dynamic programming solution for two sequences and poly-time algorithm for $O(1)$ sequences

Takeaways and open questions

Takeaways

1. Chain letter petitions exhibit one-ended growth, divergence, and mutation: intriguing reconstruction problem
2. NP-hard in general, but dynamic programming solution for two sequences and poly-time algorithm for $O(1)$ sequences
3. Efficient heuristic for more sequences

Takeaways and open questions

Takeaways

1. Chain letter petitions exhibit one-ended growth, divergence, and mutation: intriguing reconstruction problem
2. NP-hard in general, but dynamic programming solution for two sequences and poly-time algorithm for $O(1)$ sequences
3. Efficient heuristic for more sequences

Open questions

Takeaways and open questions

Takeaways

1. Chain letter petitions exhibit one-ended growth, divergence, and mutation: intriguing reconstruction problem
2. NP-hard in general, but dynamic programming solution for two sequences and poly-time algorithm for $O(1)$ sequences
3. Efficient heuristic for more sequences

Open questions

1. Approximation algorithm: bounding topological error seems hard

Takeaways and open questions

Takeaways

1. Chain letter petitions exhibit one-ended growth, divergence, and mutation: intriguing reconstruction problem
2. NP-hard in general, but dynamic programming solution for two sequences and poly-time algorithm for $O(1)$ sequences
3. Efficient heuristic for more sequences

Open questions

1. Approximation algorithm: bounding topological error seems hard
2. Efficient algorithms for small λ

Acknowledgment

Thanks to:

- Jon Kleinberg
- Anna Johnson
- Hailey Jones
- Dave Musicant
- Layla Oesper
- Anna Rafferty
- Ethan Somes

Availability

The paper is available at
`https://doi.org/10.4230/LIPIcs.CPM.2020.11`
Data and source code hosted at
`https://github.com/tomlinsonk/diverging-string-seqs`