# The tandem duplication distance is NP-hard

Manuel Lafond, Binhai Zhu, Peng Zou

UNIVERSITÉ DE SHERBROOKE

MONTANA STATE UNIVERSITY

# Tandem duplications

- Tandem duplication (**TD**)
  - String operation that copies a substring and pastes it right after.
- $AXB \rightarrow AXXB$
  - $X$ could be any substring

- e.g. abcbcabc → abcbcacbcabc

# Tandem duplication distance

- Tandem duplication *distance*
  - $d(S, T)$ = minimum number of TDs required to transform $S$ into $T$

# Tandem duplication distance

- Tandem duplication *distance*
  - $d(S, T) =$ minimum number of TDs required to transform $S$ into $T$

abc

abcabbcabcabc

# Tandem duplication distance

- Tandem duplication *distance*
  - $d(S, T) =$ minimum number of TDs required to transform $S$ into $T$

abc

abcabc

abcabbcabcabc

# Tandem duplication distance

- Tandem duplication *distance*
  - $d(S, T) =$ minimum number of TDs required to transform $S$ into $T$

abc

a<u>bca</u>bc

abcabbcabcabc

# Tandem duplication distance

- Tandem duplication *distance*
  - $d(S, T)$ = minimum number of TDs required to transform $S$ into $T$

abc

abcabc

abcabcabc

abcabbcabcabc

# Tandem duplication distance

- Tandem duplication *distance*
  - $d(S,T) =$ minimum number of TDs required to transform $S$ into $T$

abc

abcabc

a<u>bcab</u>cabc

abcabbcabcabc

# Tandem duplication distance

- Tandem duplication *distance*
  - $d(S, T)$ = minimum number of TDs required to transform $S$ into $T$

abc

abcabc

a$\underline{\text{bcab}}$cabc

a$\underline{\text{bcabbcab}}$cabc

# Tandem duplication distance

- Tandem duplication *distance*
  - $d(S,T)$ = minimum number of TDs required to transform $S$ into $T$

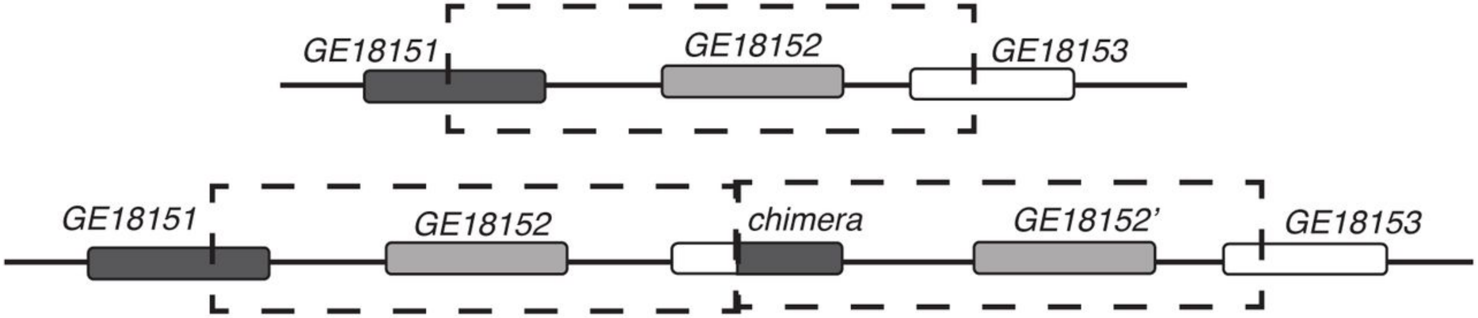
abc
abcabc
abcabcabc
abcabbcabcabc

$$d(S,T) = 3$$

(I think)

# Some history

- Extensively studied in bioinformatics
  - Most common dup mechanism [Szostak 1980]
  - Occurs in cancer [Oesper & al. 2010]
  - Gene clusters evolve by TD [Gascuel & al. 2003]



# Some history

- TD language of a string $S$
  - $td(S)$ = strings that can be generated from $S$ by TDs
- Introduced in for *copying systems*
  - [Andrzej & Rozenberg, DAM 1984]
  - If $S$ is in binary, then $td(S)$ is regular
  - Otherwise, $td(S)$ is not regular

# Some history

- TD language of a string $S$
  - $td(S)$ = strings that can be generated from $S$ by TDs
- Introduced in for *copying systems*
  - [Andrzej & Rozenberg, DAM 1984]
  - If $S$ is in binary, then $td(S)$ is regular
  - Otherwise, $td(S)$ is not regular
- Rediscovered in 2004
  - [Leupold, Mitrana & Sempere, DAM, 2004]
  - Other formal language questions studied

# Some history

- In [Leupold & al. 2004]
  - Open problem: given $S, T$, decide if $T$ is in $td(S)$.
    - Easy for binary alphabets, otherwise unknown
  - Open problem: complexity of computing $d(S, T)$
    - Unknown even on binary alphabet

# Some history

- In [Leupold & al. 2004]
  - Open problem: given $S, T$, decide if $T$ is in $td(S)$.
    - Easy for binary alphabets, otherwise unknown
  - Open problem: complexity of computing $d(S, T)$
    - Unknown even on binary alphabet
- In [Alon & al., IEEE ToIT, 2017]
  - Max value of $d(S, T)$ in terms of $|T|$ (if well-defined)
  - If $S$ is binary and square-free, then $d(S, T) \in \Theta(|T|)$
  - Also ask about the complexity of $d(S, T)$

# Our contributions

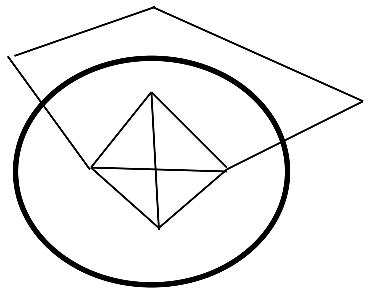
- Computing $d(S, T)$ is NP-hard.
  - Even if S is *exemplar*: has no duplicate character.
  - Solves the 15 years-old open problem of Leupold & al.
- Reduction from new NP-hard problem
  - Cost-Effective Subgraph (bonus W[1]-hardness result)
- FPT result: if $S$ is exemplar, $d(S, T)$ can be found in time $2^{O(k^2)} poly(n)$

# NP hardness

- Reduction from Cost-effective subgraph problem

# Cost effective subgraph

- Let $G$ be a graph and let $c \in \mathbb{N}$.
- For $X \subseteq V(G)$, let $E(X) = \{uv \in E(G) : u, v \in X\}$
- $cost(X) = c(|E(G)| - |E(X)|) + |X||E(X)|$

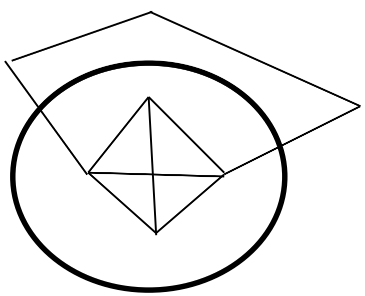


$k$-clique

$$c\left(m - \binom{k}{2}\right) + k\binom{k}{2}$$

# Cost effective subgraph

- Let $G$ be a graph and let $c \in \mathbb{N}$.
- For $X \subseteq V(G)$, let $E(X) = \{uv \in E(G): u, v \in X\}$
- $cost(X) = c(|E(G)| - |E(X)|) + |X||E(X)|$
  - We pay $c$ for each edge not in $X$, and $|X|$ for each edge inside $X$.
  - Generalization: each edge in $X$ costs $f(|X|)$.



$k$-clique

$$c\left(m - \binom{k}{2}\right) + k\binom{k}{2}$$

# Cost effective subgraph

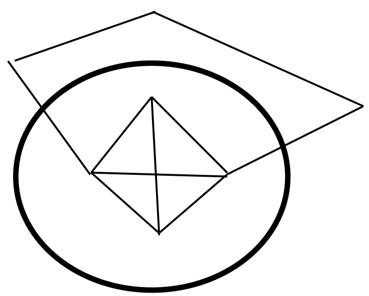
- Let $G$ be a graph and let $c \in \mathbb{N}$.
- For $X \subseteq V(G)$, let $E(X) = \{uv \in E(G) : u, v \in X\}$
- $cost(X) = c(|E(G)| - |E(X)|) + |X||E(X)|$
  - We pay $c$ for each edge not in $X$, and $|X|$ for each edge inside $X$.
  - Generalization: each edge in $X$ costs $f(|X|)$.
- Want to contain many edges, but diminishing returns on size of $X$.

# Cost effective subgraph

- Given: graph $G$, cost $c$, integer $k$
- Q: is there $X \subseteq V(G)$ such that $cost(X) \leq k$?

# Cost effective subgraph is NP-hard

- Reduction from CLIQUE.
  - Take CLIQUE instance $(G, k)$.
  - Put $c = 3k/2$
  - Graph $G$ has a clique of size $k$ if and only if $G$ has $X \subseteq V(G)$ of cost $cm - k/2 \binom{k}{2}$



$k$-clique

($\Rightarrow$)

$$c\left(m - \binom{k}{2}\right) + k\binom{k}{2} = cm - k/2\binom{k}{2}$$

($\Leftarrow$) Calculate stuff, show that only a $k$-clique can achieve this cost (not even a $(k+1)$-clique.

# Cost effective subgraph is NP-hard

- Reduction from CLIQUE.
  - Take CLIQUE instance $(G, k)$.
  - Put $c = 3k/2$
  - Graph $G$ has a clique of size $k$ if and only if $G$ has $X \subseteq V(G)$ of cost $cm - k/2 \binom{k}{2}$

- Cost-effective subgraph is W[1]-hard with respect to parameter $c$.
  - Unknown: hardness w.r.t. $cost(X)$.

# Back to our main problem

# Contractions

- The reverse of a TD is a contraction
- $AXXB \rightarrow AXB$
- Observation
  - $d(S,T) \leq k$ iff $T$ can be transformed into $S$ using $k$ contractions

abcabbcabcabc

abcabcabc

abcabc

abc

# Idea of the reduction

- Given a Cost-effective Subgraph instance $(G, c)$
- Design strings $S$ and $T$ so that

$$T = S^* E_1 E_2 \dots E_p$$

where:

- $S^*$ is generated from $S$
- Each $E_i$ is a gadget substring for edge $e_i$ of $G$
- to go from $T$ to $S$, we must:

1) contract $S^*$ to some intermediate $S'$
2) use $S'$ to contract all the $E_i$'s
3) contract $S'$ into $S$

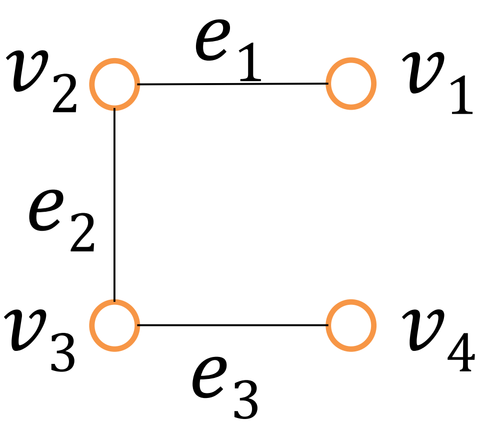
# Form of solutions

$$
\begin{aligned}
T = \; & S^* E_1 E_2 E_3 E_4 \\
& S' E_1 E_2 E_3 E_4 \\
& S' E_2 E_3 E_4 \\
& S' E_3 E_4 \\
& S' E_4 \\
& S' \\
& S
\end{aligned}
$$

# Idea of the reduction

$S = v_1 v_2 v_3 v_4 M$ ($M$ is a Mystery substring)

$T = S^* E_1 E_2 E_3$



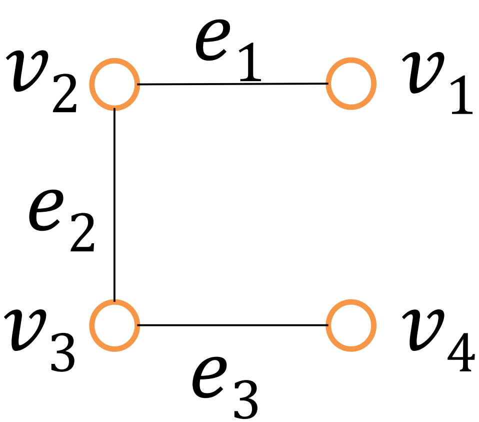
# Idea of the reduction

$S = v_1 v_2 v_3 v_4 M$ ($M$ is a Mystery substring)

$T = S^* E_1 E_2 E_3$

$S^* = v_1 v_1 v_2 v_2 v_3 v_3 v_4 v_4 M$

$E_1 = v_1 v_2 v_3 v_3 v_4 v_4 M$
$E_2 = v_1 v_1 v_2 v_3 v_4 v_4 M$
$E_3 = v_1 v_1 v_2 v_2 v_3 v_4 M$



# Idea of the reduction

$S = v_1 v_2 v_3 v_4 M$ ($M$ is a Mystery substring)

$T = S^* E_1 E_2 E_3$

$\quad S' E_1 E_2 E_3$

$S^* = v_1 v_1 v_2 v_2 v_3 v_3 v_4 v_4 M$

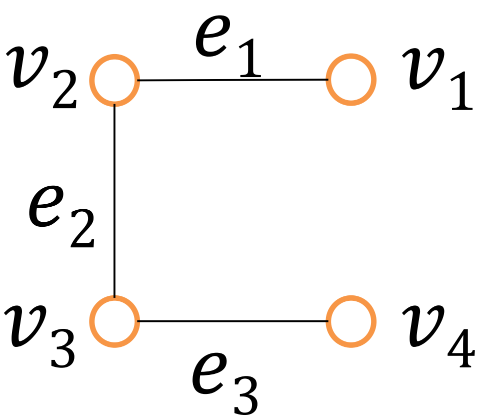
$S' = v_1 v_2 v_3 v_4 v_4 M$

represents choosing $X = \{v_1, v_2, v_3\}$

$E_1 = v_1 v_2 v_3 v_3 v_4 v_4 M$

$E_2 = v_1 v_1 v_2 v_3 v_4 v_4 M$

$E_3 = v_1 v_1 v_2 v_2 v_3 v_4 M$



# Idea of the reduction

$S = v_1 v_2 v_3 v_4 M$ ($M$ is a Mystery substring)

$T = S^* E_1 E_2 E_3$

$\boldsymbol{S' E_1} E_2 E_3$

$S^* = v_1 v_1 v_2 v_2 v_3 v_3 v_4 v_4 M$

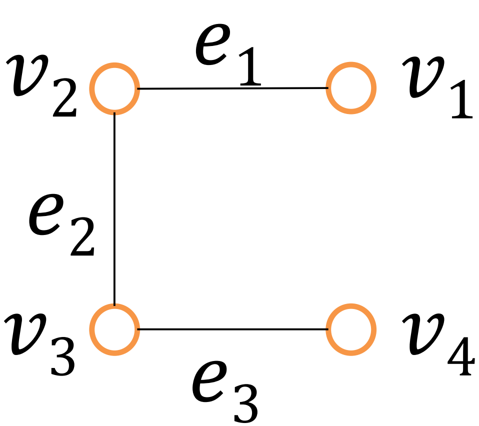
$S' = v_1 v_2 v_3 v_4 v_4 M$

represents choosing $X = \{v_1, v_2, v_3\}$

$E_1 = v_1 v_2 v_3 v_3 v_4 v_4 M$

$E_2 = v_1 v_1 v_2 v_3 v_4 v_4 M$

$E_3 = v_1 v_1 v_2 v_2 v_3 v_4 M$




Both endpoints of $E_1$ are chosen =>
$S'$ can gobble up $E_1$ in $|X| - 2$ contractions.

# Idea of the reduction

$S = v_1v_2v_3v_4M$ ($M$ is a Mystery substring)

$T = S^*E_1E_2E_3$

$S'E_1E_2E_3$

$\boldsymbol{S'E_2}E_3$

$S^* = v_1v_1v_2v_2v_3v_3v_4v_4M$

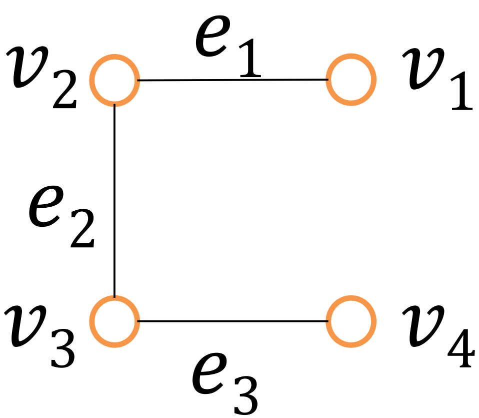
$S' = v_1v_2v_3v_4v_4M$

represents choosing $X = \{v_1, v_2, v_3\}$

$E_1 = v_1v_2v_3v_3v_4v_4M$

$E_2 = v_1v_1v_2v_3v_4v_4M$

$E_3 = v_1v_1v_2v_2v_3v_4M$




Both endpoints of $E_2$ are chosen => $S'$ can gobble up $E_2$ in $|X| - 2$ contractions.

# Idea of the reduction

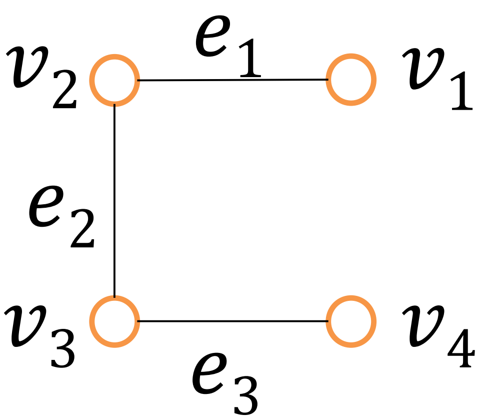
$S = v_1 v_2 v_3 v_4 M$ ($M$ is a Mystery substring)

$$T = S^* E_1 E_2 E_3$$
$$S' E_1 E_2 E_3$$
$$S' E_2 E_3$$
$$\boldsymbol{S' E_3}$$

$$S^* = v_1 v_1 v_2 v_2 v_3 v_3 v_4 v_4 M$$
$$S' = v_1 v_2 v_3 v_4 v_4 M$$

represents choosing $X = \{v_1, v_2, v_3\}$

$$E_1 = v_1 v_2 v_3 v_3 v_4 v_4 M$$
$$E_2 = v_1 v_1 v_2 v_3 v_4 v_4 M$$
$$E_3 = v_1 v_1 v_2 v_2 v_3 v_4 M$$

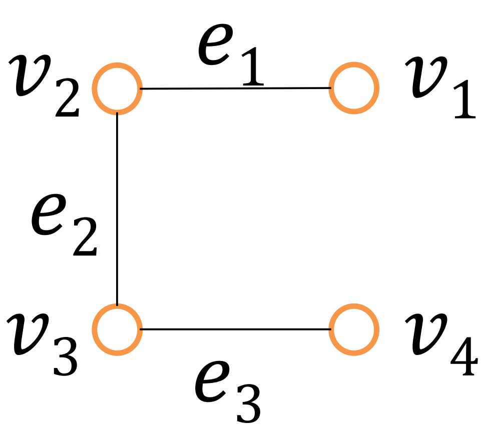



An endpoint of $E_3$ is NOT chosen => $S'$ CANNOT gobble up $E_3$. Must use mystery $M$ using ***c*** contractions.

# Idea of the reduction

$S = v_1 v_2 v_3 v_4 M$

$$T = \begin{array}{l} S^* E_1 E_2 E_3 \\ S' E_1 E_2 E_3 \\ S' E_2 E_3 \\ S' E_3 \\ S' \end{array}$$




Summary: choose intermediate $S'$ so that corresponding $X$ is such that:

- $E_i$'s contained in $X$ cost $|X| - 2$
- $E_i$'s not contained in $X$ cost $c$

Same as in cost-effective problem.

Reduction is technical to ensure that all solutions have this form.

# FPT result

**Theorem**

If $S$ is exemplar (each character occurs once), then $d(S, T)$ can be computed in time

$$2^{O(k^2)} + poly(n).$$

# FPT Result

- Idea
  - Breakpoint in $S$ = consecutive characters $xy$ in $S$ such that in $T$, either :
    - some $x$ is not followed by $y$; or
    - some $y$ is not preceded by $x$.

S = abcdefghi
T = abcdecdefghi

# FPT Result

- Idea
  - Breakpoint in $S$ = consecutive characters $xy$ in $S$ such that in $T$, either :
    - some $x$ is not followed by $y$; or
    - some $y$ is not preceded by $x$.

S = ab|cde|fghi
T = abcdecdefghi

# FPT Result

- Idea
  - Breakpoint in $S$ = consecutive characters $xy$ in $S$ such that in $T$, either :
    - some $x$ is not followed by $y$; or
    - some $y$ is not preceded by $x$.
  - A TD creates at most two breakpoints

S = ab|cde|fghi
T = abcdecdefghi

# FPT Result

- Idea
  - Breakpoint in $S$ = consecutive characters $xy$ in $S$ such that in $T$, either :
    - some $x$ is not followed by $y$; or
    - some $y$ is not preceded by $x$.
  - A TD creates at most two breakpoints
    - If $d(S,T) \leq k$, then $S$ has at most $2k$ breakpoints.

$$S = ab|cde|fghi$$
$$T = abcdecdefghi$$

## Lemma

Let $X_1, \dots, X_l$ be the maximal substrings of $S$ that have no breakpoint. Obtain $S'$ and $T'$ by replacing every occurrence of $X_1, \dots, X_l$ by a single, distinct character in $S$ and $T$.

Then $d(S,T) = d(S',T')$.

S = ab|cde|fghi
T = abcdecdefghi

S' = $X_1X_2X_3$
T' = $X_1X_2X_2X_3$

At most $2k$ breakpoints

$\Rightarrow$ $S'$ has at most $2k + 1$ characters

$\Rightarrow$ $T'$ has length at most $(2\text{k} + 1)2^k$

$\Rightarrow$ Branching over every sequence of $k$ TDs explores $O\left((k2^k)^{2k}\right)$ possibilities, hence the $2^{O(k^2)}poly(n)$ complexity.

# Summary

- Computing the TD distance is NP-hard
- Potentially useful Cost-effective subgraph problem
  - When having to balance a number of chosen elements vs diminishing returns on size
- FPT result on exemplar substrings
  - Breakpoint lemma might hold for other types of edit operations, e.g. deletions, transpositions, ...

# Open problems

- Complexity of deciding whether $S$ can generate $T$.
  - Open even for ternary alphabets.
- Complexity of $d(S, T)$ on fixed size alphabets.
  - Probably NP-hard for ternary, no idea for binary.
- FPT if $S$ is not exemplar?
- Complexity if length of a dup is bounded by a constant $c$? Is $td(S)$ context-free in this setting?