Reversal Distance for Strings with Duplicates

[1]Petr Kolman [2]Tomek Waleń

[1]Faculty of Mathematics and Physics
Charles University in Prague

[2]Wydział Matematyki, Informatyki i Mechaniki
Warsaw University

September 15, 2006

Reversal distance

reversal $\rho(i, j)$

of a string $A = a_1 \dots a_n$, $1 \le i < j \le n$, transforms the string A into a string $A' = a_1 \dots a_{i-1} a_j a_{j-1} \dots a_i a_{j+1} \dots a_n$

Reversal distance $\mathrm{RD}(A, B)$ of strings A and B

- minimum number of reversals that transform A into B

Example

$$
\begin{array}{rclll}
A & = & \mathrm{ab\underline{cccbbba}dd} & \rho(3,9) & \\
 & & \mathrm{ababbb\underline{cccdd}} & \rho(7,11) & \\
 & & \mathrm{\underline{ab}abbbddccc} & \rho(1,2) & \\
 & & \mathrm{\underline{baabbb}ddccc} & \rho(1,6) & \\
 & & \mathrm{bbbaabddccc} & = B & \Rightarrow \mathrm{RD}(A, B) = 4
\end{array}
$$

Sorting by reversals

Known results

- permutations:
  - unsigned SBR is NP-hard (Caprara 1997)
  - signed SBR is in P (Hannenhalli, Pevzner 1997)
- strings (finding the reversal distance of strings A and B):
  - SBR is NP-hard for binary strings (Christie, Irving 2001),
  - $O(\log n \log^* n)$–approximation (Cormode et al. 2002),
- strings restricted variant (k-SBR), every letter occurs at most k times,
  - $O(1)$ approximations for 2-SBR and 3-SBR (Chen et al. 2005, Chrobak et al. 2004, Goldstein et al. 2005)
  - $O(k^2)$ approximation for k-SBR (Kolman 2005)

New contribution

- $O(k)$ approximation for k-SBR in linear time

Minimum common string partition

Definitions

- *partition of a string A* - a sequence $\mathcal{P} = (P_1, P_2, \ldots, P_m)$ of strings whose concatenation is equal to A, that is $P_1 P_2 \ldots P_m = A$;
  - $P_1, P_2, \ldots, P_m$ are *blocks*
  - *size* of $\mathcal{P}$ = number of blocks
- *common partition of A and B* - a pair $(\mathcal{P}, \mathcal{Q})$ such that $\mathcal{P}$ is a partition of A, $\mathcal{Q}$ is a partition of B and $\mathcal{P}$ is a permutation of $\mathcal{Q}$
- *minimum common string partition* problem (MCSP) - find a common partition of strings A and B of minimum size

Example

$$\begin{array}{rcl} A & = & \text{abcccbbbadd} \\ B & = & \text{bbbaabddccc} \end{array}$$

Minimum common string partition

Definitions

- *partition of a string A* - a sequence $\mathcal{P} = (P_1, P_2, \ldots, P_m)$ of strings whose concatenation is equal to A, that is $P_1 P_2 \ldots P_m = A$;
  - $P_1, P_2, \ldots, P_m$ are *blocks*
  - *size* of $\mathcal{P}$ = number of blocks
- *common partition of A and B* - a pair $(\mathcal{P}, \mathcal{Q})$ such that $\mathcal{P}$ is a partition of A, $\mathcal{Q}$ is a partition of B and $\mathcal{P}$ is a permutation of $\mathcal{Q}$
- *minimum common string partition* problem (MCSP) - find a common partition of strings A and B of minimum size

Example

$$\begin{array}{rcl} A & = & \text{ab ccc bbba dd} \\ B & = & \text{bbba ab dd ccc} \end{array}$$

Minimum common string partition

Variants of MCSP

- k-MCSP - each letter occurs at most k times,
- *signed* MCSP (two blocks C and D match each other if $C = D$ or $C = -D$, where $-D$ is the reversal of D),
- the α approximation for the (signed) k-MCSP gives $O(\alpha)$ approximation for the k-SBR

A few more definitions

- *duo* - (sub)string of length two
- $duos(S)$ - the set of all duos of string S, i.e. $duos(abbab) = \{ab, ba, bb\}$,
- cutting a duo xy - cut the every occurrence of xy after the character x,

Minimum common string partition

Variants of MCSP

- *k*-MCSP - each letter occurs at most k times,
- *signed* MCSP (two blocks C and D match each other if $C = D$ or $C = -D$, where $-D$ is the reversal of D),
- the α approximation for the (signed) k-MCSP gives $O(\alpha)$ approximation for the k-SBR

A few more definitions

- *duo* - (sub)string of length two
- $duos(S)$ - the set of all duos of string S, i.e. $duos(abbab) = \{ab, ba, bb\}$,
- cutting a duo xy - cut the every occurrence of xy after the character x,

Minimum common string partition

Variants of MCSP

- *k*-MCSP - each letter occurs at most k times,
- *signed* MCSP (two blocks C and D match each other if $C = D$ or $C = -D$, where $-D$ is the reversal of D),
- the α approximation for the (signed) k-MCSP gives $O(\alpha)$ approximation for the k-SBR

A few more definitions

- *duo* - (sub)string of length two
- $duos(S)$ - the set of all duos of string S, i.e. $duos(abbab) = \{ab, ba, bb\}$,
- cutting a duo xy - cut the every occurrence of xy after the character x,

axybcdxyxybxy

Variants of MCSP

- *k-MCSP* - each letter occurs at most k times,
- *signed* MCSP (two blocks C and D match each other if $C = D$ or $C = -D$, where $-D$ is the reversal of D),
- the α approximation for the (signed) k-MCSP gives $O(\alpha)$ approximation for the k-SBR

A few more definitions

- *duo* - (sub)string of length two
- $duos(S)$ - the set of all duos of string S, i.e. $duos(abbab) = \{ab, ba, bb\}$,
- cutting a duo xy - cut the every occurrence of xy after the character x,

ax ybcdx yx ybx y

Solving MCSP

Algorithm outline

input: strings A, B

1. compute the set of the consensus duos Φ
2. $\mathcal{A}, \mathcal{B} \leftarrow$ for each duo $xy \in \Phi$, cut all occurrences of xy in A, B

output: $(\mathcal{A}, \mathcal{B})$

Example

$A = abaab \quad B = ababa$

$\Phi = \{aa,\ ba\}$ is the set of consensus duos

Solving MCSP

Algorithm outline

input: strings A, B

1. compute the set of the consensus duos Φ
2. $\mathcal{A}, \mathcal{B} \leftarrow$ for each duo $xy \in \Phi$, cut all occurrences of xy in A, B

output: $(\mathcal{A}, \mathcal{B})$

Example

$A = abaab \quad B = ababa$

$\Phi = \{aa,\ ba\}$ is the set of consensus duos

$\mathcal{A} = ab\ a\ ab$

$\mathcal{B} = ab\ ab\ a$

Solving MCSP

Algorithm outline

input: strings A, B

1. compute the set of the consensus duos Φ
2. $\mathcal{A}, \mathcal{B} \leftarrow$ for each duo $xy \in \Phi$, cut all occurrences of xy in A, B

output: $(\mathcal{A}, \mathcal{B})$

Example

$A = abaab \quad B = ababa$

$\Phi = \{aa,\ ba\}$ is the set of consensus duos

$\mathcal{A} = \{ab,\ a,\ ab\} \quad \mathcal{A}_{OPT} = \{aba,\ ab\}$
$\mathcal{B} = \{ab,\ ab,\ a\} \quad \mathcal{B}_{OPT} = \{ab,\ aba\}$

Solving MCSP – observation

Observation 1

Let #substr(A, S) - number of occurrences of substring S in string A.
If xy is a duo, such that #substr$(A, xy) \neq$ #substr(B, xy), then in every common partition of A/B, at least one occurrence of xy is cut.

Example

Observation 2

If X is a substring, such that #substr$(A, X) \neq$ #substr(B, X), then in every common partition of A/B, at least one occurrence of X is cut.

Solving MCSP – observation

Observation 1

Let #substr(A, S) - number of occurrences of substring S in string A.
If xy is a duo, such that #substr(A, xy) $\neq$ #substr(B, xy), then in every common partition of A/B, at least one occurrence of xy is cut.

Example

$$\begin{aligned} A &= \text{cbcccbccbcddd} \\ B &= \text{cdddcccbccbcb} \end{aligned}$$

Observation 2

If X is a substring, such that #substr(A, X) $\neq$ #substr(B, X), then in every common partition of A/B, at least one occurrence of X is cut.

Solving MCSP – observation

Observation 1

Let $\#\mathrm{substr}(A, S)$ - number of occurrences of substring S in string A. If xy is a duo, such that $\#\mathrm{substr}(A, xy) \neq \#\mathrm{substr}(B, xy)$, then in every common partition of A/B, at least one occurrence of xy is cut.

Example

$$\begin{aligned} A &= \text{cb cccbccb cddd} \\ B &= \text{cddd cccbccb cb} \end{aligned}$$

Observation 2

If X is a substring, such that $\#\mathrm{substr}(A, X) \neq \#\mathrm{substr}(B, X)$, then in every common partition of A/B, at least one occurrence of X is cut.

Solving MCSP – observation

Observation 1

Let $\#\text{substr}(A, S)$ - number of occurrences of substring S in string A.
If xy is a duo, such that $\#\text{substr}(A, xy) \neq \#\text{substr}(B, xy)$, then in every common partition of A/B, at least one occurrence of xy is cut.

Example

$$\begin{aligned} A &= \text{cb cccbccb cddd} \\ B &= \text{cddd cccbccb cb} \end{aligned}$$

Observation 2

If X is a substring, such that $\#\text{substr}(A, X) \neq \#\text{substr}(B, X)$, then in every common partition of A/B, at least one occurrence of X is cut.

Algorithm HS

input: strings A, B

1. construct an instance $(U, \mathcal{S})$ of the Hitting Set problem:
   $U \leftarrow duos(A) \cup duos(B)$
   $T \leftarrow \{X \mid \#\text{substr}(A, X) \neq \#\text{substr}(B, X)\}$
   $\mathcal{S} \leftarrow \{duos(X) \mid X \in T\}$
2. solve (approximately) the Minimum Hitting Set problem:
   $\Phi \leftarrow$ a hitting set for $(U, \mathcal{S})$
3. transform the hitting set into a common partition:
   $\mathcal{A}, \mathcal{B} \leftarrow$ for each duo $xy \in \Phi$, cut all occurrences of xy in A, B

output: $(\mathcal{A}, \mathcal{B})$

Algorithm HS – example

Example

$A = abaab$ $B = ababa$

$U = \{aa, ab, ba\}$
$T = \{aa, ba, aab, aba, baa, bab, abaa, abab, baba, abaab, ababa\}$
$S = \{\{aa\}, \{ba\}, \{aa, ab\}, \{aa, ba\}, \{ab, ba\}, \{aa, ab, ba\}\}$

$\Phi = \{aa,\ ba\}$ is a hitting set for $(U, \mathcal{S})$
$\mathcal{A} = \{ab,\ a,\ ab\}$
$\mathcal{B} = \{ab,\ ab,\ a\}$

Algorithm HS – correctness

Lemma

The partition $(\mathcal{A}, \mathcal{B})$ found by algorithm HS is a common partition of A, B.

Proof: by contradiction

Let #blocks$(\mathcal{P}, S)$ - num. of blocks $P_i = S$ in partition $\mathcal{P} = (P_1, \ldots, P_m)$.

Let X be the longest block s.t. #blocks$(\mathcal{A}, X) \neq$ #blocks$(\mathcal{B}, X)$

$$\#\text{blocks}(\mathcal{A}, X) = \#\text{substr}(A, X) - \sum_{Y \in \mathcal{A} : X \sqsubset Y} \#\text{substr}(Y, X) \cdot \#\text{blocks}(\mathcal{A}, Y)$$

$$\#\text{blocks}(\mathcal{B}, X) = \#\text{substr}(B, X) - \sum_{Y \in \mathcal{B} : X \sqsubset Y} \#\text{substr}(Y, X) \cdot \#\text{blocks}(\mathcal{B}, Y)$$

By choice of X: #blocks$(\mathcal{A}, Y) =$ #blocks$(\mathcal{B}, Y)$ for each Y s.t. $X \sqsubset Y$

$$\Rightarrow \#\text{substr}(A, X) \neq \#\text{substr}(B, X)$$

However, X was not cut by the algorithm – a contradiction.

Algorithm HS – efficiency

Difficulties

- The hitting set problem is NP-hard
- It is also hard to approximate (no $O(1)$–approximation).

Idea

Exploit the structure of the sets

- each set corresponds to a substring of A or B
- *"is a substring of"* defines a partial order on the set T

Approximating minimum hitting set

Minimum elements

X is a minimum element of T if no $Y \in T$ is a proper substring of X
Let $T_{\min} \leftarrow$ minimum elements of
$T = \{X \mid \#\mathrm{substr}(A, X) \neq \#\mathrm{substr}(B, X)\}$

Lemma

If $X \in T_{\min}$ then there exists an occurrence of X in A or in B that goes over cut from the optimal solution.

Proof: by contradiction: for $X \in T_{\min}$, assume that no occurrence of it in A and B goes over an optimal break
every occurrence of X in A and B is a substring of a block of the optimal partition
$\Rightarrow X$ occurres the same number of times in A and B
$\Rightarrow X \notin T \Rightarrow X \notin T_{\min}$

Approximating minimum hitting set

Procedure for hitting set

$T = \{X \mid \#\mathrm{substr}(A, X) \neq \#\mathrm{substr}(B, X)\}$
$T_{\min} \leftarrow$ minimum elements of T
$\Phi \leftarrow \emptyset$
for each $X \in T_{\min}$
  if $duos(X) \cap \Phi = \emptyset$ **then** add the first and last duo of X to Φ

Lemma

$$|\Phi| \leq 4 \cdot |OPT|$$

Proof outline

For each duo from Φ, charge some cut in the optimal solution. Each cut from the optimal solution will be charged at most 2 times.

Conclusion

Lemma

The algorithm HS *computes a* $4k$*-approximation of the minimum common partition of* A *and* B*.*

Linear-time implementation

Exploits linear-time algorithms for

- suffix trees
- special case of disjoint set union problem

Theorem

There exists an algorithm that computes in linear time $\Theta(k)$*-approximation for signed, unsigned and reversed* k-MCSP *and for signed and unsigned* k-SBR*.*

Conclusion

Results

- $O(k)$–approximation for the k–MCSP,
- the approximation for the k-SMCSP, gives the $O(k)$ approximation for the k-SBR,
- the running time $O(n)$

Challenges

- Find a better approximation, e.g., $O(\log k)$