On the Complexity of the Swap Common Superstring Problem

Paola Bonizzoni[2], Riccardo Dondi[1], Giancarlo Mauri[2], and Italo Zoppis[2]

[1] DSLCS, Università degli Studi di Bergamo, Bergamo - Italy
[2] DISCo, Università degli Studi di Milano-Bicocca, Milano - Italy
bonizzoni@disco.unimib.it, riccardo.dondi@unibg.it,
mauri@disco.unimib.it, zoppis@disco.unimib.it

Abstract. In several areas, in particular in bioinformatics, Shortest Common Superstring problem (SCS) and variants thereof have been successfully applied for strings comparison. In this paper we consider a variant of SCS that have been recently introduced, Swapped Common Superstring (SWCS), and we investigate its complexity. We show that SWCS is APX-hard even when the input strings have length bounded by a constant (equal to 10) or are over a binary alphabet.

1 Introduction

In several areas, such as bioinformatics [5], the Shortest Common Superstring problem (SCS) has been successfully applied for strings comparison [2,6,7,8]. Recently, some variants of the SCS problem have been proposed to deal with problems in bioinformatics and AI planning [4,3]. In this paper we consider one of these variants, the Swapped Common Superstring (SWCS) problem, where, given a set S of strings over an alphabet Σ and a text $\mathcal{T}$ over Σ, we look for a swap ordering $\mathcal{T}'$ of $\mathcal{T}$ (an ordering of $\mathcal{T}$ obtained by swapping only some pairs of adjacent characters) such that the number of input strings that are substrings of $\mathcal{T}'$ is maximized.

The SWCS is known to be NP-hard [4], while a relaxed version of the problem, where each occurrence of a string in the swap ordering $\mathcal{T}'$ is counted, is polynomial time solvable [4].

We investigate the complexity of SWCS under two natural parameters, i.e. the length of the input strings and the size of the alphabet. We show that SWCS is APX-hard even when the input strings have length bounded by a constant (equal to 10) or they are over a binary alphabet.

Now, we define the problem formally. Given a string s and a substring s_x of s, we say that s_x is *covered* by s.

Problem 1. [4] SWCS
Input: a set $S = \{s_1, \ldots, s_n\}$ of strings over alphabet Σ, a text $\mathcal{T} = t_1 t_2 \ldots t_m$, where each t_i, $1 \leq i \leq m$, is a character in Σ.
Output: an ordering $\mathcal{T}'$ of the text $\mathcal{T}$ (called a *swap ordering of* $\mathcal{T}$) that maximizes the number of strings in S that are covered by $\mathcal{T}'$, where $\mathcal{T}'$ is induced

by a permutation $\pi : \{1, \ldots, m\} \rightarrow \{1, \ldots, m\}$ such that: (1) if $\pi(i) = j$, then $\pi(j) = i$, (2) for all i, $\pi(i) \in \{i-1, i, i+1\}$, (3) if $\pi(i) \neq i$ then $t_{\pi(i)} \neq t_i$.

The definition given above establishes that a swap ordering $\mathcal{T}'$ of $\mathcal{T}$ is obtained by swapping only some pairs of adjacent distinct characters in $\mathcal{T}$.

2 Complexity of SWCS for Bounded Length and Alphabet

In this section, we consider two restrictions of SWCS, namely the case when the input strings length is bounded by 10 (denoted by $10-$SWCS) and the case when the input strings are over a binary alphabet (denoted by SWCS(2)). We show that both these restrictions are APX-hard.

In order to prove that $10-$SWCS and SWCS(2) are APX-hard, we present two L-reductions from the Maximum Independent Set on Cubic Graphs (MAX-ISC). We recall that a graph is cubic when each of its vertices has degree three. Given a cubic graph $G = (V, E)$, with $V = \{v_1, \ldots, v_q\}$, MAX-ISC asks for a set $V' \subseteq V$ of maximum cardinality, such that for each $v_i, v_j \in V'$, it holds $\{v_i, v_j\} \notin E$. MAX-ISC is known to be APX-hard [1].

2.1 APX-hardness of $10-$SWCS

We show that $10-$SWCS is APX-hard, giving a reduction from MAX-ISC. Let $G = (V, E)$ be a cubic graph, with $V = \{v_1, \ldots, v_q\}$. In what follows, given a vertex $v_i \in V$, denote by v_j, v_h, v_l the three vertices of G adjacent to v_i. Next, we define an instance $(S, \mathcal{T})$ of $10-$SWCS associated with G. The alphabet Σ over which the strings in S range, is defined as follows: $\Sigma = \{w_i, x_i : v_i \in V\} \cup \{a_{i,j} : \{v_i, v_j\} \in E\} \cup \{y\}$.

The set S of input strings is defined as follows: $S = \bigcup_{i:v_i \in V}(I_{i,1} \cup I_{i,2} \cup I_{i,3})$, where $I_{i,1}$, $I_{i,2}$, $I_{i,3}$, with $v_i \in V$, are three sets of strings defined as follows: $I_{i,1} = \{w_i a_{i,j} w_j, w_i a_{i,h} w_h, w_i a_{i,l} w_l\}$, $I_{i,2} = \{w_i x_i a_{i,j}, w_i x_i a_{i,h}, w_i x_i a_{i,l}\}$, $I_{i,3} = \{x_i w_i a_{i,j} w_j x_i w_i a_{i,h} w_h x_i w_i\}$.

Now, we define the text $\mathcal{T}$. For each v_i let $\mathcal{T}_i$ be the following string:

$$\mathcal{T}_i = w_i x_i a_{i,j} w_j w_i x_i a_{i,h} w_h w_i x_i a_{i,l} w_l$$

Then, $\mathcal{T}$ is defined as follows:

$$\mathcal{T} = \mathcal{T}_1 yyy \mathcal{T}_2 \ldots yyy \ldots yyy \mathcal{T}_n$$

Given a swap ordering $\mathcal{T}'$ of $\mathcal{T}$, we denote by $\mathcal{T}_i'$ the swap ordering in $\mathcal{T}'$ of the substring $\mathcal{T}_i$ of $\mathcal{T}$, with $1 \leq i \leq q$. We say that $\mathcal{T}_i'$ has a *configuration a*, if each pair w_i, x_i of $\mathcal{T}_i$ is swapped in $\mathcal{T}_i'$, that is $\mathcal{T}_i' = x_i w_i a_{i,j} w_j x_i w_i a_{i,h} w_h x_i w_i a_{i,l} w_l$. $\mathcal{T}_i'$ has a *configuration b* if no pair is swapped in $\mathcal{T}_i$, that is $\mathcal{T}_i'$ is identical to $\mathcal{T}_i$.

Next, we can show the following property of a swap ordering $\mathcal{T}'$ of $\mathcal{T}$.

Lemma 1. *Given a swap ordering $\mathcal{T}'$ of $\mathcal{T}$, we can compute in polynomial time a swap ordering $\mathcal{T}''$ of $\mathcal{T}$ such that $\mathcal{T}''$ covers at least as many input strings as*

$\mathcal{T}'$, *and such that, denoted as* $\mathcal{T}_i''$ *the swap ordering in* $\mathcal{T}''$ *of the substring* $\mathcal{T}_i$ *of* $\mathcal{T}$, *it follows that each* $\mathcal{T}_i''$ *has either a* configuration a *or a* configuration b.

Notice that a *configuration a* of ${\mathcal{T}'}_i$ allows to cover 4 input strings, namely the strings in $I_{i,1} \cup I_{i,3}$, while a *configuration b* of ${\mathcal{T}'}_i$ allows to cover 3 input strings, namely the strings in $I_{i,2}$. A *configuration a* of ${\mathcal{T}'}_i$ corresponds to a vertex v_i in an independent set of G, while a *configuration b* of ${\mathcal{T}'}_i$ corresponds to a vertex v_i in a vertex cover of G. Lemma 2 is a consequence of this observation and of Lemma 1.

Lemma 2. *Let* $G = (V, E)$ *be an instance of MAX-ISC and let* $(S, \mathcal{T})$ *be the corresponding instance of* $10 - \text{SWCS}$. *Then, there is an independent set of* G *of size* p *if and only if there is a swap ordering of* $\mathcal{T}$ *that covers* $4p + 3(q - p)$ *input strings.*

A direct consequence of Lemma 2 is the following theorem.

Theorem 1. *The* $10 - \text{SWCS}$ *problem is APX-hard.*

2.2 APX-hardness of SWCS(2)

In this section we show that SWCS(2) is APX-hard. Let $G = (V, E)$ be a cubic graph, with $V = \{v_1, \ldots, v_q\}$. For each $v_i \in V$, denote by $\{v_i, v_j\}$, $\{v_i, v_h\}$, $\{v_i, v_k\}$ the three edges incident on v_i. The input text $\mathcal{T}$ consists of $2q - 1$ substrings. More precisely, $\mathcal{T}$ contains a substring $B(v_i)$ for each $v_i \in V$. These substrings are separated by $q - 1$ identical substrings, denoted as SE:

$$\mathcal{T} = SE \cdot B(v_1) \cdot SE \cdot B(v_2) \cdot \ldots SE \cdot B(v_q) \cdot SE$$

Each substring SE is defined as follows: $SE = 1111100000$. Now, given a vertex $v_i \in V$, we define the associated substring $B(v_i)$. In order to define $B(v_i)$, we have to introduce some notations. First, given a substring s of size 5, s has an *inactive configuration*, denoted as $I(s)$, if $s = 00000$, s has a *positive configuration*, denoted as $P(s)$, if $s = 00100$, s has a *negative configuration*, denoted as $N(s)$, if $s = 01000$.

Now, we introduce three strings that will be used to define $B(v_i)$. Given a vertex $v_i \in V$, let $s(v_i)$ be a string of length 5, which can have an *inactive* configuration $I(s(v_i))$, a *positive* configuration $P(s(v_i))$ or *negative* configuration $N(s(v_i))$. Now, we define the following strings:

$$B(v_i, e_{i,j}) = I(s(v_1)) \cdot I(s((v_2)) \ldots P(s(v_i)) \ldots P(s(v_j)) \cdot \cdots \cdot I(s(v_q))$$

$$B(v_i, e_{i,h}) = I(s(v_1)) \cdot I(s(v_2)) \ldots P(s(v_i)) \ldots P(s(v_h)) \ldots I(s(v_q))$$

$$B(v_i, e_{i,l}) = I(s(v_1)) \cdot I(s(v_2)) \ldots P(s(v_i)) \ldots P(s(v_l)) \cdot \cdots \cdot I(s(v_q))$$

The substring $B(v_i)$ is defined as follows: $B(v_i) = B(v_i, e_{i,j}) \cdot B(v_i, e_{i,h}) \cdot B(v_i, e_{i,l})$.

Given a swap ordering $\mathcal{T}'$ of $\mathcal{T}$, we denote by $B'(v_i)$ ($B'(v_i, e_{i,j})$ respectively) the swap ordering of $B(v_i)$ ($B(v_i, e_{i,j})$ respectively) in $\mathcal{T}'$. A string $B'(v_i, e_{i,j})$, with $\{v_i, v_j\} \in E$, has a *positive configuration* in $\mathcal{T}'$ if both $s(v_i)$ and $s(v_j)$

have positive configurations in $\mathcal{T}'$; $B'(v_i, e_{i,j})$, with $\{v_i, v_j\} \in E$, has a *negative configuration* in $\mathcal{T}'$ if both $s(v_i)$ and $s(v_j)$ have negative configurations in $\mathcal{T}'$.

Now, we define the set S of input strings. For each edge $\{v_i, v_j\}$, S contains an input string:

$$s_{i,j} = I(s(v_1)) \dots N(s(v_i)) \dots N(s(v_j)) \cdots I(s(v_q))$$

Furthermore, for each $v_i \in V$, S contains the following two input strings $s'_i = 00000 \cdot B(v_i) \cdot SE$, $s''_i = SE \cdot B(v_i) \cdot 11111$.

First, we will show a property of the instance $(S, \mathcal{T})$ of SWCS.

Lemma 3. *Let $(S, \mathcal{T})$ be an instance of* SWCS*, then a swap ordering $\mathcal{T}'$ of $\mathcal{T}$ can cover in each substring $B'(v_i)$ either a string $s \in \{s'_i, s''_i\}$, or a string $s_{i,j}$, for some $\{v_i, v_j\} \in E$.*

Now, consider an order $\mathcal{T}'$ of $\mathcal{T}$. It can be shown that either $B'(v_i, e_{i,j})$, $B'(v_i, e_{i,h})$, $B'(v_i, e_{i,k})$ have all negative configurations (in this case $B'(v_i)$ covers the strings $s_{i,j}$, $s_{i,h}$, $s_{i,k}$ and corresponds to a vertex in an independent set of G) or $B'(v_i, e_{i,j})$, $B'(v_i, e_{i,h})$, $B'(v_i, e_{i,k})$ have all positive configurations (in this case $B'(v_i)$ covers the strings s'_i, s''_i and corresponds to a vertex in a vertex cover of G).

Lemma 4 follows from this observation and from Lemma 3.

Lemma 4. *Let $G = (V, E)$ be an instance of MAX-ISC and let $(S, \mathcal{T})$ be the corresponding instance of* SWCS(2)*. Then, there is an independent set of G of size p if and only if there is a swap ordering of $\mathcal{T}$ that covers $3p + 2(q - p)$ input strings.*

A direct consequence of Lemma 4 is the following theorem.

Theorem 2. *The* SWCS(2) *problem is APX-hard.*

References

1. P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoretical Comput. Sci.*, 237(1–2):123–134, 2000.
2. A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. *J. ACM*, 41(4):630–647, 1994.
3. R. Clifford, Z. Gotthilf, M. Lewenstein, and A. Popa. Restricted common superstring and restricted common supersequence. In *CPM 2011*, pages 467–478, 2011.
4. Z. Gotthilf, M. Lewenstein, and A. Popa. On shortest common superstring and swap permutations. In *SPIRE 2010*, pages 270–278, 2010.
5. D. Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology.* Cambridge University Press, 1997.
6. S. Ott. Lower bounds for approximating shortest superstrings over an alphabet of size 2. In *WG 1999*, pages 55–64, 1999.
7. Z. Sweedyk. A $2\frac{1}{2}$-approximation algorithm for shortest superstring. *SIAM J. Comput.*, 29(3):954–986, 1999.
8. V. Vassilevska. Explicit inapproximability bounds for the shortest superstring problem. In *MFCS 2005*, pages 793–800, 2005.