Improved Approximation Ratios for the Shortest Common Superstring Problem with Reverse Complements

Ryosuke Yamano
Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo, Japan
Division of Medical Data Informatics, Human Genome Center, Institute of Medical Science, The University of Tokyo, Japan

Tetsuo Shibuya
Division of Medical Data Informatics, Human Genome Center, Institute of Medical Science, The University of Tokyo, Japan

Abstract

The Shortest Common Superstring (SCS) problem asks for the shortest string that contains each of a given set of strings as a substring. Its reverse-complement variant, the Shortest Common Superstring problem with Reverse Complements (SCS-RC), naturally arises in bioinformatics applications, where for each input string, either the string itself or its reverse complement must appear as a substring of the superstring. The well-known MGREEDY algorithm for the standard SCS constructs a superstring by first computing an optimal cycle cover on the overlap graph and then concatenating the strings corresponding to the cycles, while its refined variant, TGREEDY, further improves the approximation ratio. Although the original 4- and 3-approximation bounds of these algorithms have been successively improved for the standard SCS, no such progress has been made for the reverse-complement setting. A previous study extended MGREEDY to SCS-RC with a 4-approximation guarantee and briefly suggested that extending TGREEDY to the reverse-complement setting could achieve a 3-approximation. In this work, we strengthen these results by proving that the extensions of MGREEDY and TGREEDY to the reverse-complement setting achieve 3.75- and 2.875-approximation ratios, respectively. Our analysis extends the classical proofs for the standard SCS to handle the bidirectional overlaps introduced by reverse complements. These results provide the first formal improvement of approximation guarantees for SCS-RC, with the 2.875-approximate algorithm currently representing the best known bound for this problem.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Shortest Common Superstring, Approximation Algorithms, DNA Sequencing

Digital Object Identifier 10.4230/LIPIcs.CPM.2026.15

Funding This work was supported by MEXT KAKENHI Grant Numbers 21H05052, 23H03345, and 23K18501.

1 Introduction

The Shortest Common Superstring (SCS) problem is a classical problem in string algorithms and combinatorial optimization. Given a set S of strings, the goal is to find a shortest possible string that contains every string in S as a substring. SCS has numerous applications in various fields [7], among which one of the most prominent is DNA sequencing. A DNA molecule consists of four nucleotides (Adenine, Thymine, Guanine, and Cytosine) and is assembled from short sequence reads. This process can be viewed as an instance of the SCS problem over a quaternary alphabet. However, unlike ordinary strings, DNA sequences are double-stranded, and sequencing technologies often produce reads whose orientation is not known, i.e., which of the two DNA-duplex strands they came from [9]. Consequently, for each


© Ryosuke Yamano and Tetsuo Shibuya;
licensed under Creative Commons License CC-BY 4.0

read, either the read itself or its reverse complement may occur in the original genome. To capture this property, the Shortest Common Superstring problem with Reverse Complements (SCS-RC) extends the classical SCS problem by requiring the superstring to contain, for each input string, either the string or its reverse complement as a substring. This extension provides a more realistic abstraction of genomic sequence reconstruction, yet its algorithmic properties and approximation guarantees have been studied far less extensively than those of the standard SCS problem.

For the standard SCS problem, Blum et al. [2] showed that the GREEDY algorithm, which repeatedly merges two distinct strings with the maximum overlap, is 4-approximate. They also introduced two variants, MGREEDY and TGREEDY, and proved that these achieve 4- and 3-approximation ratios, respectively. Many subsequent works have further improved these approximation ratios [1, 3, 14, 11, 12, 4]. Englert et al. [5] established that the MGREEDY algorithm achieves a guarantee of $\frac{\sqrt{67}+2}{3} \approx 3.396$, and by combining this with the $\frac{2}{3}$-approximation algorithm for the maximum asymmetric traveling salesman problem [10, 13], they obtained the currently best-known approximation ratio for SCS, $\frac{\sqrt{67}+14}{9} \approx 2.465$.

For the SCS-RC problem, Jiang et al. [8] extended the work of Blum et al. [2] to the reverse-complement setting. They proved that the extension of the MGREEDY algorithm, which we refer to as MGREEDY-RC (shown in Algorithm 1), is a 4-approximate algorithm for SCS-RC. They further noted that the TGREEDY algorithm can be similarly extended (the TGREEDY-RC algorithm shown in Algorithm 3) to achieve a 3-approximation, relying on the fact that the GREEDY-RC algorithm (shown in Algorithm 2) achieves a $\frac{1}{2}$ compression ratio, which was formally proved in [6].

However, no further improvements for SCS-RC have been reported since then, leaving a substantial gap compared to the standard SCS. Our analysis narrows this gap and provides new insights into greedy algorithms under the reverse-complement setting.

Our Contributions

We first establish Theorem 1, which provides a framework analogous to that used in the standard SCS problem [4, 5].

▶ **Theorem 1.** *If MGREEDY-RC is a $(2+\alpha)$-approximation algorithm and there exists an algorithm that achieves a compression ratio of δ for SCS-RC, then one can construct a $(2+(1-\delta)\alpha)$-approximation algorithm for SCS-RC. In particular, TGREEDY-RC corresponds to the case $\delta = \frac{1}{2}$ within this general framework, which immediately yields a $(2+\frac{\alpha}{2})$-approximation guarantee.*

We extend the work of Kaplan and Shafrir [11] to the reverse-complement setting, improving the approximation guarantee of MGREEDY-RC from 4 to 3.75. The presence of reverse complements prevents the direct application of their method, since it relies on the overlap rotation lemma (stated in Lemma 13), which no longer holds when the rotated string is reverse complemented. This highlights the additional structural challenges of the SCS-RC problem compared with the standard SCS problem. Nevertheless, we manage to adapt part of their improvement by exploiting the properties of reverse complements.

▶ **Theorem 2.** *MGREEDY-RC is a 3.75-approximate algorithm.*

Combining Theorem 1 and Theorem 2, we immediately obtain improved guarantees for TGREEDY-RC, tightening its approximation ratio from 3 to 2.875, which currently represents the best-known bound for SCS-RC.

▶ **Theorem 3.** *TGREEDY-RC is a 2.875-approximate algorithm.*

2 Preliminaries

Over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{t}, \mathtt{g}, \mathtt{c}\}$, we define the *complement mapping* $\bar{\cdot} : \Sigma \to \Sigma$ by

$$\bar{\mathtt{a}} = \mathtt{t}, \quad \bar{\mathtt{t}} = \mathtt{a}, \quad \bar{\mathtt{g}} = \mathtt{c}, \quad \bar{\mathtt{c}} = \mathtt{g}.$$

For a string $s = b_1 b_2 \dots b_n \in \Sigma^*$, its *reverse complement* is defined as

$$\bar{s}^R = \bar{b_n}\, \bar{b_{n-1}} \dots \bar{b_1}.$$

In other words, $\bar{s}^R$ is obtained by reversing s and then replacing each character $b_i \in \Sigma$ with its complement $\bar{b_i}$. Let $S = \{s_1, \dots, s_m\}$ be a set of strings over Σ, and define $\bar{S}^R = \{\bar{s_1}^R, \dots, \bar{s_m}^R\}$. Without loss of generality, we assume that $S \cup \bar{S}^R$ is *substring-free*, i.e., no string in $S \cup \bar{S}^R$ is a substring of another. For a string x, we use

$$x' \in \{x, \bar{x}^R\}$$

to denote either x or its reverse complement. We will use this notation consistently throughout the paper. Let us formally define the SCS-RC problem:

▶ **Definition 4** (Shortest Common Superstring with Reverse Complements (SCS-RC)).
Input: *A set of strings $S = \{s_1, \dots, s_m\}$ over an alphabet Σ.*
Output: *The shortest string s such that for each $s_i \in S$, either s_i or its reverse complement $\bar{s_i}^R$ appears as a substring of s.*

We call a (not necessarily shortest) string that contains, for each $s_i \in S$, either s_i or its reverse complement $\bar{s_i}^R$, an *approximate solution* for the SCS-RC instance S.

There are two common ways to measure the quality of approximation algorithms.

▶ **Definition 5** (Length ratio and compression ratio). *Let* $\mathrm{ALG}(S)$ *denote the length of the approximate solution for the SCS-RC instance S produced by an algorithm* ALG*, and let* $\mathrm{OPT}(S)$ *denote the length of an optimal solution. We define the* length ratio *of the algorithm as* $\frac{\mathrm{ALG}(S)}{\mathrm{OPT}(S)}$*, and the* compression ratio *as* $\frac{||S|| - \mathrm{ALG}(S)}{||S|| - \mathrm{OPT}(S)}$*, where* $||S|| = \sum_{s_i \in S} |s_i|$.

Since $\mathrm{ALG}(S) \geq \mathrm{OPT}(S)$, we distinguish between two notions of approximation. When $\delta > 1$, a δ-approximation refers to the *length ratio*, and when $\delta < 1$, it refers to the *compression ratio*.

For two (not necessarily distinct) strings x and y, let v be the longest string such that $x = uv$ and $y = vw$ for some nonempty strings u and w. We call v the *overlap* between x and y, and denote its length by $|v| = \mathrm{ov}(x, y)$. For example, if $u = \mathtt{atat}$ and $v = \mathtt{tata}$, then $\mathrm{ov}(u, v) = 3$ and $\mathrm{ov}(u, u) = 2$. The string u is called the *prefix* of x with respect to y, and is denoted by $\mathrm{pref}(x, y)$. We define the *distance* from x to y as

$$\mathrm{dist}(x, y) = |\mathrm{pref}(x, y)| = |x| - \mathrm{ov}(x, y).$$

For a given sequence of strings $x_1, \dots, x_r$, we define

$$\langle x_1, \dots, x_r \rangle = \mathrm{pref}(x_1, x_2)\mathrm{pref}(x_2, x_3) \dots \mathrm{pref}(x_{r-1}, x_r)x_r.$$

If $x_1, \dots, x_r$ are substring-free, then this string is a shortest string containing $x_1, \dots, x_r$ as substrings in this order. As observed in [2], the optimal solution of SCS-RC must have the form $\langle s'_{i_1}, \dots, s'_{i_m} \rangle$ for some permutation $i_1, \dots, i_m$ of $\{1, \dots, m\}$.

Algorithm 1 MGREEDY-RC.

Input : A set of strings S
Output : An approximate solution for the SCS-RC instance S
$T \leftarrow \emptyset$
while $S \neq \emptyset$ **do**
  $P \leftarrow \{(x, y) \in (S \cup \bar{S}^R) \times (S \cup \bar{S}^R) \mid (x = y)$ or $(x \neq y$ and $\bar{x}^R \neq y)\}$
  take (x, y) from P that maximizes the value $\mathrm{ov}(x, y)$
  if $x = y$ **then**
    $S \leftarrow S \setminus \{x, \bar{x}^R\}$
    $T \leftarrow T \cup \{x\}$
  else
    $S \leftarrow S \setminus \{x, \bar{x}^R, y, \bar{y}^R\}$
    $S \leftarrow S \cup \{\langle x, y\rangle\}$
  end
end
Concatenate all the strings in T

Algorithm 2 GREEDY-RC.

Input : A set of strings S
Output : An approximate solution for the SCS-RC instance S
while $|S| > 1$ **do**
  $P \leftarrow \{(x, y) \in (S \cup \bar{S}^R) \times (S \cup \bar{S}^R) \mid x \neq y$ and $\bar{x}^R \neq y)\}$
  take (x, y) from P that maximizes the value $\mathrm{ov}(x, y)$
  $S \leftarrow S \setminus \{x, \bar{x}^R, y, \bar{y}^R\}$
  $S \leftarrow S \cup \{\langle x, y\rangle\}$
end
Return the only element of S

The *distance graph* $G_S = (V, E, w)$ is the weighted complete directed graph constructed from the strings in $S \cup \bar{S}^R$. The vertex set is $V = S \cup \bar{S}^R$, and the edge set is $E = \{(x, y) \mid x, y \in V\}$. The weight of each edge $(x, y) \in E$ is defined as $w(x, y) = \mathrm{dist}(x, y)$. The *overlap graph* is defined analogously, except that the weight of each edge (x, y) is given by $w(x, y) = \mathrm{ov}(x, y)$. Each path $x_1, \ldots, x_r$ in G_S corresponds to the string $\langle x_1, \ldots, x_r\rangle$. Let $C = x_1, \ldots, x_r, x_1$ be a cycle in G_S. We define the *weight* of the cycle C as $w(C) = \sum_{i=1}^{r} \mathrm{dist}(x_i, x_{i+1})$, where we set $x_{r+1} = x_1$. A *cycle cover* of G_S is a set of vertex-disjoint cycles such that, for each $1 \leq i \leq m$, exactly one of s_i and ${\bar{s}_i}^R$ is contained in the cycles. A cycle cover of G_S is said to be *optimal* if it has the minimum total weight among all possible cycle covers. We denote an optimal cycle cover of G_S by $\mathrm{CYC}(G_S)$, and its total weight by $w(\mathrm{CYC}(G_S))$. Since an optimal solution to SCS-RC corresponds to a single cycle that contains exactly one of s_i and ${\bar{s}_i}^R$ for each $1 \leq i \leq m$, it can be viewed as a special case of a cycle cover. Therefore, we have the following inequation

$$w(\mathrm{CYC}(G_S)) \leq \mathrm{OPT}(S). \tag{1}$$

Algorithm 3 TGREEDY-RC.

Input : A set of strings S
Output : An approximate solution for the SCS-RC instance S
1. Compute the set T using the MGREEDY-RC algorithm
2. Apply the GREEDY-RC algorithm to T

2.1 Greedy algorithms

Jiang et al. [8] introduced a greedy algorithm shown in Algorithm 1, which we refer to as MGREEDY-RC. Strictly speaking, they considered *reversals* rather than *reverse complements*, but their formulation and analysis can be adapted directly to our setting by interpreting their s^R as $\bar{s}^R$. MGREEDY-RC disallows merging string pairs of the form $(x, \bar{x}^R)$ for $x \neq \bar{x}^R$, since both (distinct) x and $\bar{x}^R$ need not be included simultaneously. As discussed in [8], the MGREEDY-RC algorithm can be viewed as constructing a cycle cover. When it merges two strings $x = \langle s'_{i_1}, \ldots, s'_{i_h} \rangle$ and $y = \langle s'_{j_1}, \ldots, s'_{j_g} \rangle$, this corresponds to selecting an edge between s'_{i_h} and s'_{j_1} to create a new path $s'_{i_1}, \ldots, s'_{i_h}, s'_{j_1}, \ldots, s'_{j_g}$. When it merges x and $\bar{y}^R$, it first replaces the second path $s'_{j_1}, \ldots, s'_{j_g}$ with $\bar{s'_{j_g}}^R, \ldots, \bar{s'_{j_1}}^R$, and then merges the paths to form $s'_{i_1}, \ldots, s'_{i_h}, \bar{s'_{j_g}}^R, \ldots, \bar{s'_{j_1}}^R$. When MGREEDY-RC moves a string $x = \langle s'_{i_1}, \ldots, s'_{i_h} \rangle$ from S to T, it closes the corresponding path into a cycle $s'_{i_1}, \ldots, s'_{i_h}, s'_{i_1}$. These operations together form a cycle cover of G_S, and it has been shown that this cover is optimal.

▶ **Lemma 6** ([8]). *MGREEDY-RC creates an optimal cycle cover $CYC(G_S)$.*

Fici et al. [6] proved that the GREEDY-RC algorithm, given in Algorithm 2, achieves a compression ratio of $\frac{1}{2}$, and that this bound is tight for the algorithm. As in [8], they actually considered *reversals* rather than *reverse complements*, but the same analysis applies when interpreting s^R as $\bar{s}^R$.

▶ **Lemma 7** ([6]). *GREEDY-RC achieves a compression ratio of $\frac{1}{2}$.*

The TGREEDY-RC algorithm, presented in Algorithm 3, is a refinement of MGREEDY-RC. Rather than concatenating the strings in T directly, it merges them by executing GREEDY-RC.

2.2 Periodicity and the overlap rotation lemma

We adopt the notation from [3, 11]. A string x is called a *factor* of a string s if $s = x^i y$ for some positive integer i and some (possibly empty) prefix y of x. We denote by $\mathrm{factor}(s)$ the shortest such string x, and define the *period* of s as $\mathrm{period}(s) = |\mathrm{factor}(s)|$. A semi-infinite string s is called *periodic* if $s = xs$ for some nonempty string x. In this case, we denote the shortest such x by $\mathrm{factor}(s)$ and define $\mathrm{period}(s) = |\mathrm{factor}(s)|$.

Let x and y be strings that are either finite or periodic semi-infinite. We call x and y *equivalent* if $\mathrm{factor}(y)$ is a cyclic shift of $\mathrm{factor}(x)$, i.e., there exist strings e and f such that $\mathrm{factor}(x) = ef$ and $\mathrm{factor}(y) = fe$; otherwise, we call them *inequivalent*. The following lemmas were proved in [2] and restated in [3].

▶ **Lemma 8** ([2]). *Let $C = s'_{i_1}, s'_{i_2}, \ldots, s'_{i_k}, s'_{i_1}$ be a cycle in $CYC(G_S)$. Then*

$$\begin{aligned}\mathrm{factor}(\langle s'_{i_1}, \ldots, s'_{i_k}\rangle) &= \mathrm{factor}(\langle s'_{i_1}, \ldots, s'_{i_k}, s'_{i_1}\rangle)\\ &= \mathrm{pref}(s'_{i_1}, s'_{i_2}) \ldots \mathrm{pref}(s'_{i_{k-1}}, s'_{i_k})\mathrm{pref}(s'_{i_k}, s'_{i_1}),\\ \mathrm{period}(\langle s'_{i_1}, \ldots, s'_{i_k}\rangle) &= w(C).\\ \langle s'_{i_1}, \ldots, s'_{i_k}, s'_{i_1}\rangle &= \mathrm{factor}(\langle s'_{i_1}, \ldots, s'_{i_k}\rangle)s'_{i_1}\end{aligned}$$

▶ **Lemma 9** ([2]). *The strings $\langle s'_{i_1}, \ldots, s'_{i_k}\rangle$, $\langle s'_{i_2}, \ldots, s'_{i_k}, s'_{i_1}\rangle$, $\ldots$, $\langle s'_{i_k}, s'_{i_1}, \ldots, s'_{i_{k-1}}\rangle$ are all equivalent.*

▶ **Lemma 10** ([2]). *Let $C = s'_{i_1}, s'_{i_2}, \ldots, s'_{i_h}, s'_{i_1}$ and $D = s'_{j_1}, s'_{j_2}, \ldots, s'_{j_g}, s'_{j_1}$ be two distinct cycles in G_S. If $\langle s'_{i_1}, s'_{i_2}, \ldots, s'_{i_h}\rangle$ is equivalent to $\langle s'_{j_1}, s'_{j_2}, \ldots, s'_{j_g}\rangle$ Then there exists a third cycle E with weight $w(C)$ containing all the vertices in C and D.*

Lemma 10 implies that strings taken from distinct cycles in an optimal cycle cover are inequivalent, and this property naturally extends to the reverse-complement setting.

▶ **Lemma 11.** *Let $C = s'_{i_1}, s'_{i_2}, \ldots, s'_{i_h}, s'_{i_1}$ and $D = s'_{j_1}, s'_{j_2}, \ldots, s'_{j_g}, s'_{j_1}$ be two distinct cycles in $CYC(G_S)$. Then the strings $e = \langle s'_{i_1}, \ldots, s'_{i_h}\rangle$ and $f = \langle s'_{j_1}, \ldots, s'_{j_g}\rangle$ are inequivalent. Moreover, the pairs $(e, \bar{f}^R)$, $(\bar{e}^R, f)$, and $(\bar{e}^R, \bar{f}^R)$ are also inequivalent.*

The following lemma was used to gain a 4-approximate bound in [8].

▶ **Lemma 12** ([2]). *If strings x and y are inequivalent, then $\mathrm{ov}(x, y) \leq \mathrm{period}(x) + \mathrm{period}(y)$.*

Given a semi-infinite string $\alpha = x_1 x_2 \cdots$, we denote the rotation $\alpha[k] = x_k x_{k+1} \cdots$. Breslauer et al. [3] proved the following overlap rotation lemma.

▶ **Lemma 13** (overlap rotation lemma [3]). *Let α be a periodic semi-infinite string. There exists an integer k, such that for any finite string s that is inequivalent to α,*

$$\mathrm{ov}(s, \alpha[k]) \leq \mathrm{period}(s) + \frac{1}{2}\mathrm{period}(\alpha)$$

We denote a rotation $\alpha[k]$ that satisfies Lemma 13 as the *critical rotation*. The following lemma, originally proved in [3] and restated in [11], shows that it is possible to extract such a rotation from each cycle in $CYC(G_S)$. Property (4) relies on the fact that x_D and x_C are inequivalent, which follows from Property (3) together with Lemmas 9 and 11.

▶ **Lemma 14** ([3]). *Let $C = s'_{i_1}, \ldots, s'_{i_r}, s'_{i_1}$ be a cycle in $CYC(G_S)$. Then there exist a string x_C and an index j such that:*

(1) *The string $\langle s'_{i_{j+1}}, \ldots, s'_{i_r}, s'_{i_1}, \ldots, s'_{i_j}\rangle$ is a suffix of x_C.*
(2) *The string x_C is contained in $y_C = \langle s'_{i_j}, \ldots, s'_{i_r}, s'_{i_1}, \ldots, s'_{i_j}\rangle$.*
(3) *x_C is equivalent to $\langle s'_{i_{j+1}}, \ldots, s'_{i_r}, s'_{i_1}, \ldots, s'_{i_j}\rangle$.*
(4) *The semi-infinite string $\mathrm{factor}(x_C)^\infty$ is the critical rotation of $\mathrm{factor}(\langle s'_{i_1}, \ldots, s'_{i_r}\rangle)^\infty$. Specifically, let x_D be the string obtained by this lemma corresponding to a different cycle $D \in CYC(G_S)$; then $\mathrm{ov}(x_D, x_C) \leq w(D) + \frac{1}{2}w(C)$.*

3 Analysis

We begin by proving Theorem 1, which serves as the reverse-complement analogue of Theorem 3.1 in [4]. The argument follows the same ideas as the original proof.

3.1 Proof of Theorem 1

In MGREEDY-RC, we first compute the optimal cycle cover $CYC(G_S)$ and obtain the set of strings corresponding to its cycles, denoted by T (see Algorithm 1). We then observe that the optimal solution to the SCS-RC problem on this set T is at most twice as long as the optimal solution to the original instance.

▶ **Lemma 15.** $\mathrm{OPT}(T) \leq \mathrm{OPT}(S) + w(CYC(G_S)) \leq 2\,\mathrm{OPT}(S)$.

Proof. For each cycle $C = s'_{i_1}, \ldots, s'_{i_r}, s'_{i_1}$ in $CYC(G_S)$, we select a single representative string s'_{i_1}. Let $S' \subseteq S$ denote the set of these representatives, one per cycle. Let u be an optimal solution to the SCS-RC problem on S'. Clearly, $|u| \leq \mathrm{OPT}(S)$.

For each string $s'_{i_1} \in S'$, either s'_{i_1} or its reverse complement $\bar{s'_{i_1}}^R$ occurs in u. If s'_{i_1} occurs, we replace one occurrence of it with

$$e = \mathrm{pref}(s'_{i_1}, s'_{i_2}) \ldots \mathrm{pref}(s'_{i_{r-1}}, s'_{i_r})\mathrm{pref}(s'_{i_r}, s'_{i_1})s'_{i_1}.$$

Otherwise, we replace $\bar{s'_{i_1}}^R$ with $\bar{e}^R$. Note that s'_{i_1} is both the prefix and the suffix of e, and similarly $\bar{s'_{i_1}}^R$ is both the prefix and the suffix of $\bar{e}^R$. This ensures that the replacement operation is well-defined and produces a valid superstring.

This replacement increases the length of u by $w(CYC(G_S))$, yielding an approximate solution for the SCS-RC instance T. ◀

In practice, we do not know the optimal solution for the instance T. Instead, we can apply the given δ-approximation algorithm in terms of the compression ratio. Let $\mathrm{ALG}(T)$ denote the length of the solution produced by this δ-approximation algorithm on the set T. By the definition of the compression ratio, we have

$$\delta\,(||T|| - \mathrm{OPT}(T)) \leq ||T|| - \mathrm{ALG}(T),$$

where $||T|| = \sum_{x \in T} |x|$ denotes the total length of strings in T. Note that $||T||$ is also the length of the superstring produced by MGREEDY-RC. Hence, the $(2+\alpha)$-approximation guarantee of MGREEDY implies

$$||T|| \leq (2+\alpha)\mathrm{OPT}(S).$$

Combining these inequalities, we obtain

$$\begin{aligned}\mathrm{ALG}(T) &\leq (1-\delta)||T|| + \delta\,\mathrm{OPT}(T)\\ &\leq (1-\delta)(2+\alpha)\mathrm{OPT}(S) + 2\delta\,\mathrm{OPT}(S)\\ &= (2+(1-\delta)\alpha)\mathrm{OPT}(S).\end{aligned}$$

TGREEDY-RC corresponds to the case where we run GREEDY-RC on the set T. From Lemma 7, GREEDY-RC is a $\frac{1}{2}$-approximation algorithm in terms of compression ratio, that is, $\delta = \frac{1}{2}$.

3.2 Proof of Theorem 2

For each cycle $C = s'_{i_1}, s'_{i_2}, \ldots, s'_{i_r}, s'_{i_1}$ in $CYC(G_S)$, the existence of strings x_C and y_C satisfying the properties stated in Lemma 14 is guaranteed. We define

$$A = \{x_C \mid C \in CYC(G_S)\}, \qquad ||A|| = \sum_{x \in A} |x|.$$

By property (1) of Lemma 14, $||A||$ provides an upper bound on $||T||$.

▶ **Lemma 16.** $||T|| \leq ||A||$.

Proof. Consider a cycle $C = s'_{i_1}, s'_{i_2}, \ldots, s'_{i_r}, s'_{i_1}$ in $CYC(G_S)$, and let j be the index whose existence is guaranteed by Lemma 14. By property (1) of that lemma, we have

$$|\langle s'_{i_{j+1}}, \ldots, s'_{i_r}, s'_{i_1}, \ldots, s'_{i_j}\rangle| \leq |x_C|.$$

Since MGREEDY-RC greedily merges string pairs with the maximum overlap, the cycle-closing edge from s'_{i_r} to s'_{i_1} has the smallest overlap among edges in the cycle. Hence,

$$\mathrm{ov}(s'_{i_r}, s'_{i_1}) \leq \mathrm{ov}(s'_{i_k}, s'_{i_{k+1}}) \quad \text{for all } 1 \leq k < r.$$

The string produced by MGREEDY-RC is $\langle s'_{i_1}, \ldots, s'_{i_r}\rangle$, whose length can be computed as

$$\begin{aligned}|\langle s'_{i_1}, \ldots, s'_{i_r}\rangle| &= |\mathrm{pref}(s'_{i_1}, s'_{i_2})\mathrm{pref}(s'_{i_2}, s'_{i_3}) \ldots \mathrm{pref}(s'_{i_{r-1}}, s'_{i_r})s'_{i_r}| \\ &= |\mathrm{pref}(s'_{i_1}, s'_{i_2})\mathrm{pref}(s'_{i_2}, s'_{i_3}) \ldots \mathrm{pref}(s'_{i_{r-1}}, s'_{i_r})\mathrm{pref}(s'_{i_r}, s'_{i_1})| + \mathrm{ov}(s'_{i_r}, s'_{i_1}) \\ &= w(C) + \mathrm{ov}(s'_{i_r}, s'_{i_1}).\end{aligned}$$

On the other hand, for the rotated sequence starting at index $j+1$, we have

$$\begin{aligned}|\langle s'_{i_{j+1}}, \ldots, s'_{i_r}, s'_{i_1}, \ldots, s'_{i_j}\rangle| &= |\mathrm{pref}(s'_{i_{j+1}}, s'_{i_{j+2}}) \ldots \mathrm{pref}(s'_{i_{r-1}}, s'_{i_r})\mathrm{pref}(s'_{i_r}, s'_{i_1})| \\ &\quad + |\mathrm{pref}(s'_{i_1}, s'_{i_2}) \ldots \mathrm{pref}(s'_{i_j}, s'_{i_{j+1}})| + \mathrm{ov}(s'_{i_j}, s'_{i_{j+1}}) \\ &= w(C) + \mathrm{ov}(s'_{i_j}, s'_{i_{j+1}}).\end{aligned}$$

Since $\mathrm{ov}(s'_{i_r}, s'_{i_1}) \leq \mathrm{ov}(s'_{i_j}, s'_{i_{j+1}})$, we conclude that

$$|\langle s'_{i_1}, \ldots, s'_{i_r}\rangle| \leq |\langle s'_{i_{j+1}}, \ldots, s'_{i_r}, s'_{i_1}, \ldots, s'_{i_j}\rangle|.$$

Summing over all cycles in $CYC(G_S)$ yields $||T|| \leq ||A||$. ◀

From the previous lemma, we have established that $||T|| \leq ||A||$. Our next goal is to obtain an upper bound on $||A||$. To this end, consider the SCS-RC problem on A, and let $\mathrm{OV}(A)$ denote the maximal overlap in A, defined as

$$\mathrm{OV}(A) = \sum_{j=1}^{k-1} \mathrm{ov}(x'_{C_j}, x'_{C_{j+1}}),$$

where $C_1, \ldots, C_k$ are the $k = |CYC(G_S)|$ cycles in $CYC(G_S)$, ordered according to an optimal superstring $\langle x'_{C_1}, \ldots, x'_{C_k}\rangle$ for the SCS-RC instance on A.

Then, by definition, we have

$$||A|| = \mathrm{OPT}(A) + \mathrm{OV}(A).$$

An upper bound on $\mathrm{OPT}(A)$ can be obtained in a manner similar to Lemma 15.

▶ **Lemma 17.** $\mathrm{OPT}(A) \leq \mathrm{OPT}(S) + w(CYC(G_S))$.

Proof. Let $B = \{y_C \mid C \in CYC(G_S)\}$. By property (2) of Lemma 14, each y_C contains x_C, which implies $\mathrm{OPT}(A) \leq \mathrm{OPT}(B)$.

Let $y_C = \langle s'_{i_j}, \ldots, s'_{i_r}, s'_{i_1}, \ldots, s'_{i_j}\rangle$, and select s'_{i_j} as the representative of the cycle C. Let $S' \subseteq S$ denote the set of these representatives over all cycles $C \in CYC(G_S)$.

Let u be an optimal solution of SCS-RC for S'. Clearly, $|u| \leq \mathrm{OPT}(S)$. For each string $s'_{i_j} \in S'$, either s'_{i_j} or its reverse complement $\bar{s'_{i_j}}^R$ occurs in u. If s'_{i_j} occurs, replace once occurence of it with the original y_C; otherwise, replace $\bar{s'_{i_j}}^R$ with $\bar{y_C}^R$. This replacement increases the length of u by $w(CYC(G_S))$, yielding an approximate solution for the SCS-RC instance B. Hence, $\mathrm{OPT}(B) \leq \mathrm{OPT}(S) + w(CYC(G_S))$. Together with $\mathrm{OPT}(A) \leq \mathrm{OPT}(B)$, we obtain $\mathrm{OPT}(A) \leq \mathrm{OPT}(S) + w(CYC(G_S))$. ◀

For each cycle $C = s'_{i_1}, s'_{i_2}, \ldots, s'_{i_r}, s'_{i_1}$ in $CYC(G_S)$, we can consider its reverse complement, $\bar{C}^R = \bar{s'_{i_1}}^R, \bar{s'_{i_r}}^R, \ldots, \bar{s'_{i_2}}^R, \bar{s'_{i_1}}^R$. For this $\bar{C}^R$, there exists a string $x_{\bar{C}^R}$ that satisfies the properties stated in Lemma 14. However, since $x_{\bar{C}^R}$ and $\bar{x_C}^R$ may differ, we cannot directly derive the upper bound $\mathrm{OV}(A) \leq 1.5\, w(CYC(G_S))$ from Lemmas 13 and 14.

Therefore, when the reverse complement $\bar{x_C}^R$ appears in the optimal solution, we must apply Lemma 12 instead of Lemma 13. The key observation is that, by also considering the reverse complement of the entire optimal superstring, the proportion of such strings can be limited to at most half.

▶ **Lemma 18.** $\mathrm{OV}(A) \leq 1.75\, w(CYC(G_S))$.

Proof. Let $C_1, \ldots, C_k$ be the $k = |CYC(G_S)|$ cycles in $CYC(G_S)$, ordered according to an optimal solution for the SCS-RC instance on A. Let $v = \langle x'_{C_1}, \ldots, x'_{C_k} \rangle$ denote this optimal superstring. Recall that for any $i \neq j$, the strings x'_{C_i} and x'_{C_j} are inequivalent, as discussed in Section 2.2. Then,

$$\begin{aligned}\mathrm{OV}(A) &= \sum_{j=1}^{k-1} \mathrm{ov}(x'_{C_j}, x'_{C_{j+1}}) \\ &\leq \sum_{j=2}^{k} \begin{cases} w(C_{j-1}) + 0.5\, w(C_j), & \text{if } x'_{C_j} = x_{C_j} \quad \text{(by Lemma 13)}, \\ w(C_{j-1}) + w(C_j), & \text{if } x'_{C_j} = \bar{x_{C_j}}^R \quad \text{(by Lemma 12)}. \end{cases}\end{aligned}$$

Rearranging the terms gives

$$\mathrm{OV}(A) \leq \sum_{j=1}^{k} \begin{cases} 1.5\, w(C_j), & \text{if } x'_{C_j} = x_{C_j}, \\ 2\, w(C_j), & \text{if } x'_{C_j} = \bar{x_{C_j}}^R. \end{cases}$$

Next, consider the reverse complement of the optimal solution, $\bar{v}^R = \langle \bar{x'_{C_k}}^R, \ldots, \bar{x'_{C_1}}^R \rangle$, which is also an optimal superstring for the same instance. In this reversed solution, every occurrence of x_{C_j} is replaced by $\bar{x_{C_j}}^R$, and vice versa. Hence, we also have

$$\mathrm{OV}(A) \leq \sum_{j=1}^{k} \begin{cases} 2\, w(C_j), & \text{if } x'_{C_j} = x_{C_j}, \\ 1.5\, w(C_j), & \text{if } x'_{C_j} = \bar{x_{C_j}}^R. \end{cases}$$

By adding the two inequalities and dividing by two, we obtain

$$2\,\mathrm{OV}(A) \leq \sum_{j=1}^{k} 3.5\, w(C_j), \quad \text{that is,} \quad \mathrm{OV}(A) \leq 1.75\, w(CYC(G_S)).$$ ◀

From Lemmas 17 and 18, we obtain the following upper bound on $||A||$:

$$\begin{aligned}||A|| &= \mathrm{OPT}(A) + \mathrm{OV}(A) \\ &\leq \mathrm{OPT}(S) + w(CYC(G_S)) + 1.75\, w(CYC(G_S)) \\ &= \mathrm{OPT}(S) + 2.75\, w(CYC(G_S)).\end{aligned}$$

Using Lemma 16, we then have

$$||T|| \leq ||A|| \leq \mathrm{OPT}(S) + 2.75\, w(CYC(G_S)) \leq 3.75\, \mathrm{OPT}(S),$$

which completes the proof of Theorem 2.

4 Conclusion and Future Work

We have improved the approximation guarantees of MGREEDY-RC and TGREEDY-RC for the SCS-RC problem by leveraging the overlap rotation lemma and certain properties of reverse complements. The remaining gaps compared to the standard SCS problem are mainly in two aspects:

(1) The approximation guarantee of MGREEDY-RC.
(2) The compression ratio.

For the first aspect, we improved the approximation guarantee from 4 to 3.75. However, for the standard SCS problem, the best known ratio of the corresponding MGREEDY algorithm is $\frac{\sqrt{67}+2}{3} \approx 3.396$ [5], so closing this gap remains an open research direction.

For the second aspect, we used the $\frac{1}{2}$-approximation compression ratio algorithm in GREEDY-RC. In the case of the standard SCS, the problem can be reduced to the maximum asymmetric traveling salesman problem (MAX-ATSP), which is known to have a $\frac{2}{3}$-approximation algorithm [10, 13]. However, for the SCS-RC problem, we must consider clusters containing two vertices each, which prevents the direct application of approximation algorithms for MAX-ATSP. Another research direction is to find a suitable reduction related to the MAX-ATSP problem, or to consider an entirely different approach to improve the compression ratio.

References

1 Chris Armen and Clifford Stein. A 2 2/3-approximation algorithm for the shortest superstring problem. In Dan Hirschberg and Gene Myers, editors, *Combinatorial Pattern Matching*, pages 87–101, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. doi:10.1007/3-540-61258-0_8.

2 Avrim Blum, Tao Jiang, Ming Li, John Tromp, and Mihalis Yannakakis. Linear approximation of shortest superstrings. *J. ACM*, 41(4):630–647, 1994. doi:10.1145/179812.179818.

3 Dany Breslauer, Tao Jiang, and Zhigen Jiang. Rotations of periodic strings and short superstrings. *J. Algorithms*, 24(2):340–353, 1997. doi:10.1006/jagm.1997.0861.

4 Matthias Englert, Nicolaos Matsakis, and Pavel Veselý. Improved approximation guarantees for shortest superstrings using cycle classification by overlap to length ratios. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 317–330, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520001.

5 Matthias Englert, Nicolaos Matsakis, and Pavel Veselý. Approximation Guarantees for Shortest Superstrings: Simpler and Better. In Satoru Iwata and Naonori Kakimura, editors, *34th International Symposium on Algorithms and Computation (ISAAC 2023)*, volume 283 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:17, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ISAAC.2023.29.

6 Gabriele Fici, Tomasz Kociumaka, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. On the greedy algorithm for the shortest common superstring problem with reversals. *Information Processing Letters*, 116(3):245–251, 2016. doi:10.1016/j.ipl.2015.11.015.

7 Theodoros P. Gevezes and Leonidas S. Pitsoulis. *The Shortest Superstring Problem*, pages 189–227. Springer New York, New York, NY, 2014. doi:10.1007/978-1-4939-0808-0_10.

8 Tao Jiang, Ming Li, and Ding-Zhu Du. A note on shortest superstrings with flipping. *Information Processing Letters*, 44(4):195–199, 1992. doi:10.1016/0020-0190(92)90084-9.

9 Eugene W. Myers Jr. A history of dna sequence assembly. *it - Information Technology*, 58(3):126–132, 2016. doi:10.1515/itit-2015-0047.

10 Haim Kaplan, Moshe Lewenstein, Nira Shafrir, and Maxim Sviridenko. Approximation algorithms for asymmetric tsp by decomposing directed regular multigraphs. *J. ACM*, 52(4):602–626, 2005. doi:10.1145/1082036.1082041.

11 Haim Kaplan and Nira Shafrir. The greedy algorithm for shortest superstrings. *Information Processing Letters*, 93(1):13–17, 2005. doi:10.1016/j.ipl.2004.09.012.

12 Marcin Mucha. *Lyndon Words and Short Superstrings*, pages 958–972. Proceedings of the 2013 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2013. doi:10.1137/1.9781611973105.69.

13 Katarzyna Paluch, Khaled Elbassioni, and Anke van Zuylen. Simpler Approximation of the Maximum Asymmetric Traveling Salesman Problem. In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, volume 14 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 501–506, Dagstuhl, Germany, 2012. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.STACS.2012.501.

14 Z. Sweedyk. A $2\frac{1}{2}$-approximation algorithm for shortest superstring. *SIAM Journal on Computing*, 29(3):954–986, 2000. doi:10.1137/S0097539796324661.