Improved Approximation Algorithms and Hardness Results for Shortest Common Superstring 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 is a fundamental task in sequence analysis. In genome assembly, however, the double-stranded nature of DNA implies that each fragment may occur either in its original orientation or as its reverse complement. This motivates the Shortest Common Superstring with Reverse Complements (SCS-RC) problem, which asks for a shortest string that contains, for each input string, either the string itself or its reverse complement as a substring. The previously best-known approximation ratio for SCS-RC was $\frac{23}{8}$. In this paper, we present a new approximation algorithm achieving an improved ratio of $\frac{8}{3}$. Our approach computes an optimal constrained cycle cover by reducing the problem, via a novel gadget construction, to a maximum-weight perfect matching in a general graph. We also investigate the computational hardness of SCS-RC. While the decision version is known to be NP-complete, no explicit inapproximability results were previously established. We show that the hardness of SCS carries over to SCS-RC through a polynomial-time reduction, implying that it is NP-hard to approximate SCS-RC within a factor better than $\frac{333}{332}$. Notably, this hardness result holds even for the DNA alphabet.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

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

Digital Object Identifier 10.4230/LIPIcs.WABI.2026.2

Related Version *Previous Version*: `https://arxiv.org/abs/2603.26176`

Funding This work was supported by MEXT KAKENHI Grant Number 23K28035.

1 Introduction

The Shortest Common Superstring (SCS) problem is to find a shortest string that contains every string in a given set as a substring. It is a classical problem in combinatorial optimization and has numerous applications across a wide range of fields [6]. A particularly important application arises in DNA assembly. A DNA molecule consists of four nucleotides, namely adenine, thymine, guanine, and cytosine. The sequencing task aims to reconstruct the original molecule from a collection of short reads, which can be naturally viewed as an instance of the SCS problem over a quaternary alphabet.

However, the standard SCS formulation does not capture a crucial aspect of genome assembly, namely the double-stranded nature of DNA. Each read may originate from either strand of the DNA molecule and may therefore appear in its original orientation or as its reverse complement. Since sequencing technologies typically do not reveal the strand of origin, the target genome must contain, for each read, either the read itself or its reverse

complement as a substring [8]. This observation motivates the *Shortest Common Superstring with Reverse Complements* (SCS-RC) problem, which seeks a shortest string that contains, for each input string, either the string itself or its reverse complement.

Over the past decades, the standard SCS problem has been extensively studied, resulting in a rich body of literature, including a wide range of approximation algorithms [2, 1, 3, 14, 9, 12, 4, 5] as well as several inapproximability results [15, 10]. The currently best-known approximation ratio guarantee is $\frac{\sqrt{67}+14}{9} \approx 2.465$, due to Englert et al. [5], whereas the strongest inapproximability result establishes that it is NP-hard to approximate SCS within a factor better than $\frac{333}{332}$ [10].

In contrast, despite its clear relevance to genome assembly, the SCS-RC problem has received considerably less attention. Jiang et al. [7] extended the work of Blum et al. [2] and presented a 3-approximation algorithm for SCS-RC. Yamano and Shibuya [16] subsequently improved the approximation ratio to $\frac{23}{8} = 2.875$, which remains the best known. On the hardness side, Kececioglu [11] proved the NP-completeness of the decision version of SCS-RC, which is, to the best of our knowledge, the only known hardness result. These results highlight a substantial gap between the current understanding of SCS and SCS-RC. In this work, we address this gap by improving both the approximation ratio guarantees and the hardness results for SCS-RC.

1.1 Our Contributions

We first establish Theorem 1, improving the approximation ratio for SCS-RC from $\frac{23}{8} = 2.875$ to $\frac{8}{3} \approx 2.67$. Our result extends the classical $\frac{8}{3}$-approximation algorithm for SCS due to Breslauer et al. [3], which is based on computing a minimum-weight cycle cover, that is, a collection of vertex-disjoint directed cycles covering all vertices of a weighted directed graph encoding string overlaps in the SCS setting. In our setting, this notion is modified as follows: instead of requiring the cycle cover to include all vertices, we require that exactly one vertex is selected from each complementary pair.

Unlike in the classical SCS setting, the presence of reverse complements introduces an additional selection constraint. This constraint prevents a direct formulation as a standard cycle cover problem and precludes a straightforward reduction to weighted bipartite matching.

More generally, consider a variant of the cycle cover problem in which the vertex set is partitioned into several arbitrarily given disjoint subsets, and one is required to select exactly one vertex from each subset. The minimum-weight cycle cover problem under this constraint is known to be NP-hard [13]. Consequently, our constrained cycle cover problem cannot be reduced to this more general variant in a straightforward manner.

Although closely related cycle cover formulations have been studied in the context of reverse complements, they typically permit self-loops, which enables simple greedy algorithms [7, 16]. In contrast, excluding self-loops is essential in our setting and makes these approaches inapplicable.

We overcome these difficulties by introducing a novel gadget construction that reduces the resulting constrained minimum-weight cycle cover problem to a weighted perfect matching problem in a general graph.

▶ **Theorem 1.** *The SCS-RC problem admits an $\frac{8}{3}$-approximation algorithm.*

On the hardness side, we show that approximation hardness for SCS transfers to SCS-RC.

▶ **Theorem 2.** *If SCS-RC can be approximated within a factor of β, then SCS can also be approximated within a factor of β.*

We further show that this hardness result persists even when the alphabet size is restricted to four, corresponding to the DNA alphabet $\Sigma_{\mathrm{DNA}} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, where ($\mathtt{A}$, $\mathtt{T}$) and ($\mathtt{C}$, $\mathtt{G}$) are complementary pairs.

▶ **Theorem 3.** *If SCS-RC can be approximated within a factor of β on instances over the DNA alphabet, then SCS-RC can also be approximated within a factor of β on instances over an arbitrary alphabet.*

By combining Theorems 2 and 3 with the known hardness result that it is NP-hard to approximate SCS within a factor better than $\frac{333}{332}$ [10], we obtain Corollary 4.

▶ **Corollary 4.** *It is NP-hard to approximate SCS-RC within a factor better than $\frac{333}{332}$. This holds even for the DNA alphabet.*

2 Preliminaries

Throughout this paper, unless stated otherwise, we consider strings over a finite alphabet Σ equipped with an involutive complement mapping $cm\colon \Sigma \to \Sigma$, that is, $cm(cm(x)) = x$ for every $x \in \Sigma$. A prominent example is the DNA alphabet $\Sigma_{\mathrm{DNA}} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, where the complement mapping is given by $cm(\mathtt{A}) = \mathtt{T}$ and $cm(\mathtt{C}) = \mathtt{G}$, reflecting the complementarity of DNA nucleotides.

For a string $s = s_1 s_2 \cdots s_n \in \Sigma^n$, its *reverse complement*, denoted by $\mathrm{rc}(s)$, is defined as

$$\mathrm{rc}(s) = cm(s_n)cm(s_{n-1}) \cdots cm(s_1).$$

This general setting allows us to capture not only the standard reverse-complement operation on DNA, but also the special case where $cm(x) = x$ for all $x \in \Sigma$, in which case $\mathrm{rc}(s)$ coincides with the usual reversal of s.

For a set of strings T, we define $\mathrm{rc}(T) = \{\mathrm{rc}(t) \mid t \in T\}$. Without loss of generality, we assume that the input set S satisfies that $S \cup \mathrm{rc}(S)$ is *substring-free*, that is, no string in $S \cup \mathrm{rc}(S)$ is a substring of another string in the same set. For a string x, we write $x' \in \{x, \mathrm{rc}(x)\}$ to represent x or its reverse complement $\mathrm{rc}(x)$. The choice of x' depends on the particular solution under consideration. Let us formally define the *Shortest Common Superstring with Reverse Complements (SCS-RC)* problem.

▶ **Definition 5** (Shortest Common Superstring with Reverse Complements (SCS-RC)).
Input: *A set of strings $S = \{s_1, \ldots, s_m\}$ over Σ such that $S \cup \mathrm{rc}(S)$ is substring-free.*
Output: *A shortest string s such that for every $s_i \in S$, either s_i or its reverse complement $\mathrm{rc}(s_i)$ appears as a substring of s.*

2.1 The Distance Graph and Cycle Cover

We follow the standard notions of *overlap*, *prefix*, and *distance* as introduced in [2]. For two strings s and t, let y be the longest string such that $s = xy$ and $t = yz$ for some non-empty strings x and z. The length of y is called the *overlap* of s and t and is denoted by $\mathrm{ov}(s, t)$. The string x is called the *prefix* of s with respect to t and is denoted by $\mathrm{pref}(s, t)$. The length of x is called the *distance* from s to t and is denoted by $\mathrm{dist}(s, t)$.

For an ordered sequence of strings $x_1, \ldots, x_r$, we define

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

Since $S \cup \mathrm{rc}(S)$ is substring-free, there exists an optimal solution to the SCS-RC problem of the form $s = \langle s'_{q_1}, \ldots, s'_{q_m} \rangle$, where $(q_1, \ldots, q_m)$ is a permutation of $\{1, \ldots, m\}$, as observed in [2] and extended to the reverse-complement setting in [7]. Let $\mathrm{OPT}(S)$ denote the length of an optimal solution.

We consider the *distance graph* induced by the input set of strings S and denote it by $G_{\mathrm{dist}}(S)$. Formally, $G_{\mathrm{dist}}(S) = (V, E, w)$ is a directed weighted graph defined as follows. The vertex set is $V = S \cup \mathrm{rc}(S)$, and the edge set is

$$E = (V \times V) \setminus \bigcup_{x \in V} \{(x, x), (x, \mathrm{rc}(x))\}.$$

Each edge $(x, y) \in E$ is assigned weight $w(x, y) = \mathrm{dist}(x, y)$.

Thus, $G_{\mathrm{dist}}(S)$ contains neither self-loops nor edges between a string and its reverse complement. This definition differs from existing distance graph constructions that take reverse complements into account [7, 16], in which self-loops are allowed. By replacing the distance function with the overlap function, we obtain the *overlap graph* $G_{\mathrm{ov}}(S)$.

For a cycle c in a distance graph, let $w(c)$ denote the weight of the cycle, defined as the sum of the weights of its edges.

Given an instance S, consider the distance graph $G_{\mathrm{dist}}(S)$. Let $\mathrm{GTSP}(G_{\mathrm{dist}}(S))$ denote the minimum weight of a directed cycle that contains exactly one vertex from each pair $\{s_i, \mathrm{rc}(s_i)\}$ for $1 \le i \le m$. This can be viewed as a special case of the generalized traveling salesman problem (GTSP) [13], where the clusters are $\{\{s_i, \mathrm{rc}(s_i)\} \mid 1 \le i \le m\}$. Then,

$$\mathrm{GTSP}(G_{\mathrm{dist}}(S)) \le \mathrm{OPT}(S),$$

and thus provides a lower bound for the SCS-RC problem. Since GTSP is NP-hard [13], we consider a relaxation of this formulation, namely the *cycle cover* problem.

A *cycle cover* of $G_{\mathrm{dist}}(S)$ is a collection of vertex-disjoint directed cycles that contains exactly one vertex from each pair $\{s_i, \mathrm{rc}(s_i)\}$ for $1 \le i \le m$. This is a constrained variant of the classical cycle cover problem. We denote by $\mathrm{CYC}(G_{\mathrm{dist}}(S))$ a minimum-weight cycle cover, and by $w(\mathrm{CYC}(G_{\mathrm{dist}}(S)))$ its total weight. Clearly,

$$w(\mathrm{CYC}(G_{\mathrm{dist}}(S))) \le \mathrm{GTSP}(G_{\mathrm{dist}}(S)) \le \mathrm{OPT}(S).$$

A similar notion applies to the overlap graph $G_{\mathrm{ov}}(S)$. In this case, one considers a maximum-weight cycle cover, which corresponds exactly to the minimum-weight cycle cover in the distance graph $G_{\mathrm{dist}}(S)$. This correspondence follows from the relation $\mathrm{ov}(x, y) = |x| - \mathrm{dist}(x, y)$, which holds for any edge from x to y in the graph.

2.2 Periodicity of Strings

We follow the definitions in [3]. A string x is a *factor* of a finite string s if $s = x^i y$ for some integer $i \ge 1$ and some (possibly empty) prefix y of x. Let $\mathrm{factor}(s)$ be the shortest such x, and define $\mathrm{period}(s) = |\mathrm{factor}(s)|$. A *periodic semi-infinite* string s is an infinite string satisfying $s = xs$ for some nonempty string x. The shortest such x is denoted by $\mathrm{factor}(s)$, and $\mathrm{period}(s) = |\mathrm{factor}(s)|$. Let s and t be strings, each of which is either finite or periodic semi-infinite. We say that s and t are *equivalent* if $\mathrm{factor}(s)$ and $\mathrm{factor}(t)$ are cyclic shifts of each other; that is, there exist strings a and b such that $\mathrm{factor}(s) = ab$ and $\mathrm{factor}(t) = ba$. Otherwise, s and t are said to be *inequivalent*. This defines an equivalence relation on such strings. In particular, equivalent strings have the same period.

3 Useful Lemmas from Previous Work

Lemma 6 relates the periodicity of overlapped strings to cycles in the distance graph. This result was proved in [2] and restated in [3].

▶ **Lemma 6** ([2]). *Let* $c = (s'_{i_1}, \ldots, s'_{i_r}, s'_{i_1})$ *be a cycle in* $\mathrm{CYC}(G_{\mathrm{dist}}(S))$. *Then*

$$w(c) = \mathrm{dist}(s'_{i_1}, s'_{i_2}) + \cdots + \mathrm{dist}(s'_{i_{r-1}}, s'_{i_r}) + \mathrm{dist}(s'_{i_r}, s'_{i_1}) = \mathrm{period}(\langle s'_{i_1}, \ldots, s'_{i_r}\rangle).$$

Moreover, the strings $\langle s'_{i_1}, \ldots, s'_{i_r}\rangle, \langle s'_{i_2}, \ldots, s'_{i_r}, s'_{i_1}\rangle, \ldots, \langle s'_{i_r}, s'_{i_1}, \ldots, s'_{i_{r-1}}\rangle$ *are all equivalent.*

The inequivalence of strings extracted from distinct cycles of a minimum-weight cycle cover was originally shown in [2], and later extended to the reverse-complement setting in [7, 16].

▶ **Lemma 7** ([2, 7]). *Let* $c = (s'_{i_1}, \ldots, s'_{i_r}, s'_{i_1})$ *and* $d = (s'_{j_1}, \ldots, s'_{j_k}, s'_{j_1})$ *be two distinct cycles in* $\mathrm{CYC}(G_{\mathrm{dist}}(S))$. *Define* $e = \langle s'_{i_1}, \ldots, s'_{i_r}\rangle$ *and* $f = \langle s'_{j_1}, \ldots, s'_{j_k}\rangle$. *Then the four pairs* (e, f), $(e, \mathrm{rc}(f))$, $(\mathrm{rc}(e), f)$, $(\mathrm{rc}(e), \mathrm{rc}(f))$ *are all inequivalent.*

Given a semi-infinite string $\alpha = a_1 a_2 \cdots$, we denote by $\alpha[k] = a_k a_{k+1} \cdots$ its rotation starting at position k. Breslauer et al. [3] proved the following overlap rotation lemma.

▶ **Lemma 8** (Overlap Rotation Lemma [3]). *Let* α *be a periodic semi-infinite string. Then there exists an integer* k *such that, for any finite string* s *that is inequivalent to* α,

$$\text{if } \mathrm{period}(s) \le \mathrm{period}(\alpha), \text{ then } \mathrm{ov}(s, \alpha[k]) \le \frac{2}{3}(\mathrm{period}(s) + \mathrm{period}(\alpha)).$$

We refer to a rotation $\alpha[k]$ obtained from Lemma 8 as the *critical rotation*. Lemma 5.1 of [3] was later restated in the reverse-complement setting as Lemma 14 in [16]. The inequalities in property (4) follow from the fact that, for distinct cycles c and d, the corresponding strings t_c and t_d are inequivalent from property (3) and Lemma 7, while $\mathrm{period}(t_c) = w(c)$ and $\mathrm{period}(t_d) = w(d)$ from property (3) and Lemma 6.

▶ **Lemma 9** ([3, 16]). *Let* $c = (s'_{i_1}, \ldots, s'_{i_r}, s'_{i_1})$ *be a cycle in* $\mathrm{CYC}(G_{\mathrm{dist}}(S))$. *Then there exist a string* t_c *and an index* j *such that the following properties hold:*

(1) *The string* $\langle s'_{i_{j+1}}, \ldots, s'_{i_r}, s'_{i_1}, \ldots, s'_{i_j}\rangle$ *is a suffix of* t_c.
(2) *The string* t_c *is a substring of* $\langle s'_{i_j}, \ldots, s'_{i_r}, s'_{i_1}, \ldots, s'_{i_j}\rangle$.
(3) *The string* t_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* $\mathrm{factor}(t_c)^\infty$ *is the critical rotation of* $\mathrm{factor}(\langle s'_{i_1}, \ldots, s'_{i_r}\rangle)^\infty$.

Moreover, let t_d *be the string obtained from this lemma corresponding to a distinct cycle* $d \in \mathrm{CYC}(G_{\mathrm{dist}}(S))$, *where* $w(d) \le w(c)$. *Then the following inequations hold:*

$$\mathrm{ov}(t_d, t_c) \le \frac{2}{3}(w(d) + w(c)), \quad \mathrm{ov}(\mathrm{rc}(t_d), t_c) \le \frac{2}{3}(w(d) + w(c)).$$

Furthermore, the string t_c *can be computed in time linear in* $w(c)$.

Lemma 2.6 of [9] was adapted to the reverse-complement setting in Lemma 17 of [16].

▶ **Lemma 10** ([9, 16]). *Let* $T = \{t_c \mid c \in \mathrm{CYC}(G_{\mathrm{dist}}(S))\}$, *where each string* t_c *is obtained as in Lemma 9. Then*

$$\mathrm{OPT}(T) \le \mathrm{OPT}(S) + w(\mathrm{CYC}(G_{\mathrm{dist}}(S))).$$

4 Approximation Algorithm

We begin by presenting an overview in Algorithm 1, which follows the framework of [3].

4.1 Overview of the Entire $\frac{8}{3}$-Approximation Algorithm

In Step 2 of Algorithm 1, the string t_j contains, for each vertex of the cycle c_j, the string corresponding to that vertex, by Property (1) of Lemma 9. In Step 4, the resulting string is $\langle t'_{i_p}, \ldots, t'_{i_r}, t'_{i_1}, \ldots, t'_{i_{p-1}} \rangle$ if $t'_{i_p} = t_{i_p}$, and $\langle t'_{i_{p+1}}, \ldots, t'_{i_r}, t'_{i_1}, \ldots, t'_{i_p} \rangle$ otherwise. In both cases, the resulting string contains, for each vertex of the cycle d, the string corresponding to that vertex. Together, these two observations ensure that Algorithm 1 outputs a valid approximate solution of SCS-RC for the instance S.

In Step 4, we break each cycle with particular care. Specifically, we remove an edge whose overlap is bounded as guaranteed by Property (4) of Lemma 9, thereby ensuring that the increase in length incurred by breaking the cycle is properly controlled. Observe that the length of the resulting string in Step 4 is $w(d) + \mathrm{ov}(t'_{i_{p-1}}, t'_{i_p})$ when $t'_{i_p} = t_{i_p}$, and $w(d) + \mathrm{ov}(t'_{i_p}, t'_{i_{p+1}})$ otherwise.

Note that self-loops are not permitted in this step. Accordingly, self-loops are excluded in our definition of distance graphs, and hence do not appear in the minimum-weight cycle covers. The algorithmic details of computing such cycle covers are deferred to Section 4.2.

In the remainder of this subsection, we assume that the minimum-weight cycle cover can be computed in polynomial time, and focus on the approximation analysis. Specifically, we show that Algorithm 1 achieves an approximation ratio of $\frac{8}{3}$.

▶ **Theorem 11.** *Algorithm 1 yields an $\frac{8}{3}$-approximation for the SCS-RC problem.*

Proof. Let $\mathrm{ALG}(S)$ denote the length of the string produced by Algorithm 1. By Step 4 of the algorithm, the length $\mathrm{ALG}(S)$ can be expressed as

$$\mathrm{ALG}(S) = w(\mathrm{CYC}(G_{\mathrm{dist}}(T))) + \sum_{d \in \mathrm{CYC}(G_{\mathrm{dist}}(T))} OV_d,$$

where OV_d denotes the overlap lost when breaking the cycle d. Combining Lemma 10 with the inequality $w(\mathrm{CYC}(G_{\mathrm{dist}}(T))) \le \mathrm{OPT}(T)$, we obtain

$$w(\mathrm{CYC}(G_{\mathrm{dist}}(T))) \le \mathrm{OPT}(S) + w(\mathrm{CYC}(G_{\mathrm{dist}}(S))).$$

We next bound the term OV_d for each cycle d. Consider first the case where $t'_{i_p} = t_{i_p}$ in Step 4. For convenience, we define $t'_{i_0} = t'_{i_r}$ and $c_{i_0} = c_{i_r}$. In this case, the overlap lost is $\mathrm{ov}(t'_{i_{p-1}}, t'_{i_p}) = \mathrm{ov}(t'_{i_{p-1}}, t_{i_p})$. By Lemma 9, we have

$$\mathrm{ov}(t'_{i_{p-1}}, t_{i_p}) \le \frac{2}{3}\big(w(c_{i_{p-1}}) + w(c_{i_p})\big) \le \frac{2}{3} \sum_{\{j \mid t'_j \in d\}} w(c_j).$$

Now consider the case where $t'_{i_p} = \mathrm{rc}(t_{i_p})$. We define $t'_{i_{r+1}} = t'_{i_1}$ and $c_{i_{r+1}} = c_{i_1}$. The overlap lost in this case is $\mathrm{ov}(t'_{i_p}, t'_{i_{p+1}}) = \mathrm{ov}(\mathrm{rc}(t_{i_p}), t'_{i_{p+1}})$. Again by Lemma 9, we obtain

$$\mathrm{ov}(\mathrm{rc}(t_{i_p}), t'_{i_{p+1}}) = \mathrm{ov}(\mathrm{rc}(t'_{i_{p+1}}), t_{i_p}) \le \frac{2}{3}\big(w(c_{i_{p+1}}) + w(c_{i_p})\big) \le \frac{2}{3} \sum_{\{j \mid t'_j \in d\}} w(c_j).$$

In both cases, the overlap lost for cycle d is bounded by $\frac{2}{3}\sum_{\{j \mid t'_j \in d\}} w(c_j)$. Summing over all cycles yields

$$\sum_{d \in \mathrm{CYC}(G_{\mathrm{dist}}(T))} OV_d \le \frac{2}{3} \sum_{d \in \mathrm{CYC}(G_{\mathrm{dist}}(T))} \sum_{\{j \mid t'_j \in d\}} w(c_j) = \frac{2}{3} w(\mathrm{CYC}(G_{\mathrm{dist}}(S))).$$

Algorithm 1 $\frac{8}{3}$-approximation algorithm for SCS-RC.

1. Construct the distance graph $G_{\mathrm{dist}}(S)$ from the input string set S, and compute a minimum-weight cycle cover $\mathrm{CYC}(G_{\mathrm{dist}}(S)) = \{c_1, \ldots, c_k\}$.
2. For each $j \in \{1, \ldots, k\}$, let t_j be the string obtained from the cycle c_j as described in Lemma 9, and let $T = \{t_1, \ldots, t_k\}$. Construct the distance graph $G_{\mathrm{dist}}(T)$.
3. Compute a minimum-weight cycle cover $\mathrm{CYC}(G_{\mathrm{dist}}(T))$.
4. For each cycle $d = (t'_{i_1}, \ldots, t'_{i_r}, t'_{i_1})$ in $\mathrm{CYC}(G_{\mathrm{dist}}(T))$, let $p \in \{1, \ldots, r\}$ be an index such that t'_{i_p} has maximum period among the vertices of d. If $t'_{i_p} = t_{i_p}$, break the cycle by deleting the edge incoming to t'_{i_p}; otherwise, delete the edge outgoing from t'_{i_p}. This produces a superstring containing all strings corresponding to the vertices of d.
5. Concatenate the resulting strings in an arbitrary order.

Combining the above bounds, we conclude that

$$\mathrm{ALG}(S) \leq \mathrm{OPT}(S) + \frac{5}{3} w(\mathrm{CYC}(G_{\mathrm{dist}}(S))) \leq \frac{8}{3}\mathrm{OPT}(S),$$

which completes the proof. ◀

4.2 Computing the Constrained Optimal Cycle Cover

We consider computing a maximum-weight cycle cover in the overlap graph, rather than a minimum-weight cycle cover in the distance graph. As discussed in Section 2.1, these two formulations are equivalent. However, the overlap graph admits the symmetry

$$\mathrm{ov}(s, t) = \mathrm{ov}(\mathrm{rc}(t), \mathrm{rc}(s))$$

for any strings s and t, which simplifies the presentation of our algorithm. In the remainder of this subsection, we fix an input set of strings S and consider the problem of computing a maximum-weight cycle cover in the overlap graph $G_{\mathrm{ov}}(S)$.

Overview of the cycle cover construction

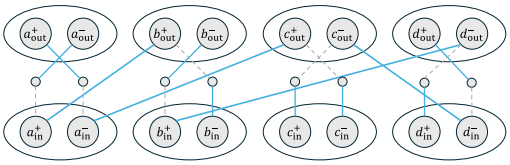
Our algorithm does not compute a cycle cover satisfying all constraints in a single step; instead, it proceeds in two stages.

In the first stage, we impose only the constraint that, for any pair of strings s and t, the edges $s \to t$ and $\mathrm{rc}(t) \to \mathrm{rc}(s)$ cannot be selected simultaneously. This constraint is enforced via a gadget construction, under which we compute a maximum-weight perfect matching. From the resulting matching, we derive a second, symmetric matching by replacing each edge $s \to t$ with its reverse-complement counterpart $\mathrm{rc}(t) \to \mathrm{rc}(s)$. The union of these two matchings admits the extraction of a maximum-weight cycle cover satisfying the constraints, together with its reverse-complement image. The overall procedure is illustrated in Figure 1.

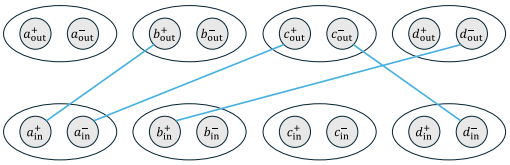
Construction of the auxiliary graph G' with gadgets

For technical convenience, we work with oriented copies of strings. For each string $s \in S$, we introduce two formal symbols s^+ and s^-, representing s and its reverse complement, respectively. Let

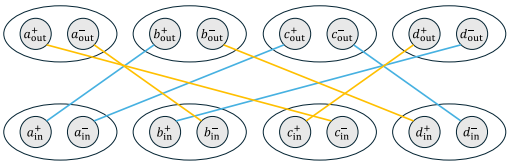
$$\tilde{S} = \bigcup_{s \in S} \{s^+, s^-\}.$$



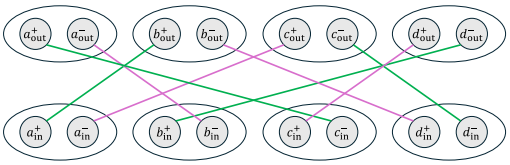
(a) Compute a maximum-weight perfect matching in G'. The small vertices in the center correspond to auxiliary vertices. The matching is shown in light blue. Dashed gray edges represent edges incident to auxiliary vertices that are not included in the matching.



(b) Remove all edges incident to auxiliary vertices to obtain the edge set F.



(c) Compute the reverse-complement image $\bar{F}$, shown in yellow.



(d) Extract two matchings M and $\bar{M}$ corresponding to the resulting maximum-weight cycle cover. Matching M, shown in green, corresponds to the cycle $(a, \mathrm{rc}(c), \mathrm{rc}(d), b, a)$, whereas matching $\bar{M}$, shown in purple, corresponds to the cycle $(\mathrm{rc}(a), \mathrm{rc}(b), d, c, \mathrm{rc}(a))$.

Figure 1 An example illustrating the computation of a maximum-weight cycle cover in an overlap graph induced by the string set $S = \{a, b, c, d\}$.

Even if $s = \mathrm{rc}(s)$, the two elements s^+ and s^- are treated as distinct vertices. This convention allows us to uniformly model the problem as selecting exactly one representative from each pair $\{s^+, s^-\}$, regardless of whether $s = \mathrm{rc}(s)$ or not.

For $u = s^+$ and $v = s^-$, we define $\bar{u} = v$ and $\bar{v} = u$. More generally, for any $u \in \tilde{S}$, we denote by $\bar{u}$ the oriented copy corresponding to the reverse complement. We also associate each $u \in \tilde{S}$ with a string $\mathrm{str}(u)$, defined by

$$\mathrm{str}(s^+) = s \quad \text{and} \quad \mathrm{str}(s^-) = \mathrm{rc}(s).$$

We recall the standard reduction from cycle covers to perfect matchings. Given a directed graph, a cycle cover can be computed by reducing the problem to a perfect matching in a bipartite graph, where each vertex is split into an outgoing and an incoming copy.

As in this reduction, each vertex $u \in \tilde{S}$ is split into two vertices u_{out} and u_{in}, representing outgoing and incoming edges, respectively. However, unlike the standard setting, we treat the two vertices $\{u, \bar{u}\}$ as a single cluster.

Formally, we first define a bipartite graph $\tilde{G} = (V_{\mathrm{out}}, V_{\mathrm{in}}, \tilde{E}, w)$ constructed from the overlap graph $G_{\mathrm{ov}}(S)$. The vertex sets are defined as

$$V_{\mathrm{out}} = \{u_{\mathrm{out}} \mid u \in \tilde{S}\} \quad \text{and} \quad V_{\mathrm{in}} = \{u_{\mathrm{in}} \mid u \in \tilde{S}\}.$$

The edge set $\tilde{E}$ is defined by

$$\tilde{E} = (V_{\mathrm{out}} \times V_{\mathrm{in}}) \setminus \bigcup_{u \in \tilde{S}} \{(u_{\mathrm{out}}, u_{\mathrm{in}}), (\bar{u}_{\mathrm{out}}, \bar{u}_{\mathrm{in}})\}.$$

For each edge $(u_{\mathrm{out}}, v_{\mathrm{in}}) \in \tilde{E}$, the weight is defined as $w(u_{\mathrm{out}}, v_{\mathrm{in}}) = \mathrm{ov}(\mathrm{str}(u), \mathrm{str}(v))$.

A matching in $\tilde{G}$ that corresponds to a valid cycle cover of the overlap graph $G_{\mathrm{ov}}(S)$ must satisfy the following condition. For each $u \in \tilde{S}$, exactly one of the following two cases holds:

(i) u_{out} and u_{in} are matched, and $\bar{u}_{\mathrm{out}}$ and $\bar{u}_{\mathrm{in}}$ are unmatched;
(ii) $\bar{u}_{\mathrm{out}}$ and $\bar{u}_{\mathrm{in}}$ are matched, and u_{out} and u_{in} are unmatched.

Case (i) corresponds to selecting the string $\mathrm{str}(u)$, whereas case (ii) corresponds to selecting its reverse complement $\mathrm{str}(\bar{u})$. Enforcing this constraint directly within a matching formulation, however, turns out to be technically challenging.

Instead, we impose a weaker constraint: for each $u \in \tilde{S}$, the vertices u_{out} and $\bar{u}_{\mathrm{in}}$ cannot be matched simultaneously. To enforce this condition, we construct a graph G' by augmenting $\tilde{G}$ with the following gadget. For each $u \in \tilde{S}$, we introduce an auxiliary vertex x_u and add two zero-weight edges (u_{out}, x_u) and $(x_u, \bar{u}_{\mathrm{in}})$.

Obtaining the edge sets F and $\bar{F}$ from a maximum-weight perfect matching in G'

The graph G' may no longer be bipartite due to the introduction of auxiliary gadgets. Accordingly, we compute a maximum-weight perfect matching in G' using an algorithm for general graphs. From the obtained matching, we discard all edges incident to auxiliary vertices and denote the remaining edge set by $F \subseteq \tilde{E}$. An example of this procedure is shown in Figures 1a and 1b. Since the matching is computed under a relaxation of the constraints required for a valid cycle cover, the total weight of F is at least that of a maximum-weight cycle cover of $G_{\mathrm{ov}}(S)$.

The constraints enforced by the gadgets ensure that, for each $u \in \tilde{S}$, edges outgoing from u_{out} and edges incoming to $\bar{u}_{\mathrm{in}}$ are not selected simultaneously in F. This implies that, for any distinct $u, v \in \tilde{S}$, the two edges $(u_{\mathrm{out}}, v_{\mathrm{in}})$ and $(\bar{v}_{\mathrm{out}}, \bar{u}_{\mathrm{in}})$ cannot be selected simultaneously. Consequently, we can construct the reverse-complement image of F by replacing each edge $(u_{\mathrm{out}}, v_{\mathrm{in}}) \in F$ with the edge $(\bar{v}_{\mathrm{out}}, \bar{u}_{\mathrm{in}})$. We denote the resulting edge set by $\bar{F}$. An example of this construction is shown in Figure 1c. By the symmetry of the overlap weights, namely $\mathrm{ov}(\mathrm{str}(u), \mathrm{str}(v)) = \mathrm{ov}(\mathrm{str}(\bar{v}), \mathrm{str}(\bar{u}))$, the total weight of $\bar{F}$ is equal to that of F. Observe that the union $F \cup \bar{F}$ is a perfect matching of $\tilde{G}$.

Extracting two optimal cycle covers from $F \cup \bar{F}$

We prove that there exists a matching M in $\tilde{G}$ corresponding to a maximum-weight cycle cover of the overlap graph $G_{\mathrm{ov}}(S)$ such that $F \cup \bar{F} = M \cup \bar{M}$, where $\bar{M}$ denotes the reverse-complement image of M. An example illustrating the extraction of M and $\bar{M}$ is shown in Figure 1d. Since $F \cup \bar{F}$ is a perfect matching of $\tilde{G}$, it induces a collection of vertex-disjoint directed cycles covering all vertices of $\tilde{S}$. Moreover, the total weight of $F \cup \bar{F}$ is an upper bound on the total weight of $M \cup \bar{M}$, since F is obtained from a relaxation of the constraints, as discussed above. Therefore, it suffices to show that each such cycle is valid with respect to the definition of a cycle cover in the original overlap graph $G_{\mathrm{ov}}(S)$.

▶ **Lemma 12.** *No cycle induced by $F \cup \bar{F}$ contains both a vertex $u \in \tilde{S}$ and its reverse-complement counterpart $\bar{u}$.*

Proof. Suppose, for the sake of contradiction, that there exists a directed cycle

$$c = (v_1, \ldots, \bar{v}_1, \ldots, v_1)$$

that contains both a vertex $v_1 \in \tilde{S}$ and its reverse-complement counterpart $\bar{v}_1$. Let $v_i \to v_{i+1}$ be an arbitrary directed edge of c.

By construction, for every edge $(e_{\mathrm{out}}, f_{\mathrm{in}}) \in F \cup \bar{F}$, its reverse-complement image $(\bar{f}_{\mathrm{out}}, \bar{e}_{\mathrm{in}})$ also belongs to $F \cup \bar{F}$. Therefore, whenever the cycle c contains the edge $v_i \to v_{i+1}$ and also contains the vertex $\bar{v}_i$, it must contain the edge $\bar{v}_{i+1} \to \bar{v}_i$ as well.

Consider traversing the cycle c simultaneously from v_1 forward and from $\bar{v}_1$ backward. Since both v_1 and $\bar{v}_1$ lie on the same directed cycle, the two traversals must eventually meet. At the meeting point, there exists an index i such that both v_i and its reverse-complement $\bar{v}_i$ appear consecutively on c, implying that c contains a directed edge of the form $v_i \to \bar{v}_i$.

However, such an edge cannot exist: by definition of $\tilde{G}$, the edge set $\tilde{E}$ contains no edge connecting a vertex to its reverse complement. This contradiction completes the proof. ◀

By Lemma 12, for each cycle c induced by $F \cup \bar{F}$, its reverse-complement image $\bar{c}$ is also induced by $F \cup \bar{F}$ and forms a distinct vertex-disjoint cycle. Hence, every induced cycle belongs to a unique pair $\{c, \bar{c}\}$, and from each pair, we can arbitrarily choose exactly one cycle. The selected cycles form a collection of vertex-disjoint directed cycles that covers exactly one vertex from each pair $\{u, \bar{u}\}$ for $u \in \tilde{S}$, and hence yields a valid cycle cover of $G_{\text{ov}}(S)$.

To conclude this section, we have shown that a maximum-weight cycle cover of the overlap graph $G_{\text{ov}}(S)$ can be computed in polynomial time. Using the equivalence between the maximum-weight cycle cover of $G_{\text{ov}}(S)$ and the minimum-weight cycle cover of $G_{\text{dist}}(S)$, this establishes Theorem 13.

▶ **Theorem 13.** *Given a distance graph $G_{\text{dist}}(S)$, the minimum-weight cycle cover $\text{CYC}(G_{\text{dist}}(S))$ can be computed in polynomial time.*

Combining Theorem 13 with Theorem 11, we obtain a polynomial-time $\frac{8}{3}$-approximation algorithm for SCS-RC. This completes the proof of our first main result, Theorem 1 from Section 1.1.

5 Hardness Results

In this section, we establish the hardness results stated in Theorems 2 and 3 from Section 1.1. We restate the theorems here for ease of reference.

▶ **Theorem 2.** *If SCS-RC can be approximated within a factor of β, then SCS can also be approximated within a factor of β.*

▶ **Theorem 3.** *If SCS-RC can be approximated within a factor of β on instances over the DNA alphabet, then SCS-RC can also be approximated within a factor of β on instances over an arbitrary alphabet.*

We first prove Theorem 2 by presenting a reduction from SCS to SCS-RC that preserves the optimal solution length. We then prove Theorem 3 by presenting a ratio-preserving reduction from SCS-RC over an arbitrary alphabet to SCS-RC over the DNA alphabet.

5.1 Proof of Theorem 2

Let $S = \{s_1, \ldots, s_m\}$ be an arbitrary instance of SCS, and let $A = \{a_1, \ldots, a_k\}$ denote the alphabet consisting of all characters appearing in the strings of S. From this instance, we construct a corresponding instance of SCS-RC as follows. Define a new alphabet

$$B = \{a_1, \ldots, a_k, a_{k+1}, \ldots, a_{2k}\},$$

and introduce a complement mapping $cm : B \to B$ defined by

$$cm(a_i) = \begin{cases} a_{i+k}, & \text{if } 1 \le i \le k, \\ a_{i-k}, & \text{if } k+1 \le i \le 2k. \end{cases}$$

It is immediate that cm is an involution on B.

Using the alphabet B equipped with the complement mapping cm, we consider the SCS-RC instance with the same input set S. We show that any approximate solution to this SCS-RC instance can be transformed into a solution that uses only strings from S, without increasing its length.

Let t be an arbitrary approximate solution to the constructed SCS-RC instance. There exists a permutation $(i_1, \ldots, i_m)$ of $\{1, \ldots, m\}$ such that t can be written as

$$t = \langle s'_{i_1}, \ldots, s'_{i_m} \rangle.$$

We define a set of intervals I as follows:

$$I = \{[l, r] \mid 1 \le l \le r \le m, \text{ and } (l = 1 \text{ or } s'_{i_{l-1}} = s_{i_{l-1}}), \text{ and } (r = m \text{ or } s'_{i_{r+1}} = s_{i_{r+1}}), \\ \text{and } s'_{i_k} = \mathrm{rc}(s_{i_k}) \text{ for all } k \in \{l, \ldots, r\}\}.$$

That is, each interval $[l, r] \in I$ corresponds to a maximal substring of reverse-complemented strings in t. For each interval $[l, r] \in I$, the string t contains the substring

$$x = \langle \mathrm{rc}(s_{i_l}), \ldots, \mathrm{rc}(s_{i_r}) \rangle.$$

We replace this substring by its reverse complement

$$\mathrm{rc}(x) = \langle s_{i_r}, \ldots, s_{i_l} \rangle.$$

Since the complement mapping cm maps symbols in A to symbols in $B \setminus A$ and vice versa, no overlap can occur between strings from S and strings from $\mathrm{rc}(S)$. Therefore, this transformation does not decrease any existing overlap, and hence does not increase the total length of the superstring. By repeatedly applying this operation to all intervals in I, we eventually obtain a solution that consists solely of strings from S, and whose length is no greater than that of the original solution.

In particular, any optimal solution to the original SCS instance is also an optimal solution to the constructed SCS-RC instance. Moreover, any β-approximate solution to the constructed SCS-RC instance can be transformed into a solution to the original SCS instance of the same length. Hence, a β-approximation algorithm for SCS-RC yields a β-approximation algorithm for SCS.

5.2 Proof of Theorem 3

Let $S = \{s_1, \ldots, s_m\}$ be an arbitrary instance of SCS-RC, and let $A = \{a_1, \ldots, a_k\}$ be the alphabet consisting of all characters appearing in the strings of $S \cup \mathrm{rc}(S)$, equipped with a complement mapping $cm \colon A \to A$. Since cm is an involution, for each $a \in A$, either $cm(a) = a$ or $cm(a) \neq a$ and $cm(cm(a)) = a$. Without loss of generality, we may relabel the symbols in A so that there exists an index j with $0 \le j \le \lfloor \frac{k}{2} \rfloor$ such that

$$cm(a_i) = \begin{cases} a_{i+j} & \text{if } 1 \le i \le j, \\ a_{i-j} & \text{if } j+1 \le i \le 2j, \\ a_i & \text{if } 2j+1 \le i \le k. \end{cases}$$

Here, we define a morphism $h \colon A^* \to \Sigma^*_{\mathrm{DNA}}$. First, for each symbol $a_i \in A$, let

$$h(a_i) = \begin{cases} \mathtt{A}^i (\mathtt{AG})^{(k+1-i-j)} \mathtt{G}^i & \text{if } 1 \le i \le j, \\ \mathtt{C}^{(i-j)} (\mathtt{CT})^{(k+1-i)} \mathtt{T}^{(i-j)} & \text{if } j+1 \le i \le 2j, \\ \mathtt{A}^{(i-j)} (\mathtt{AT})^{(k+1-i)} \mathtt{T}^{(i-j)} & \text{if } 2j+1 \le i \le k, \end{cases}$$

and extend h to A^* by concatenation, i.e., $h(xy) = h(x)h(y)$ for all $x, y \in A^*$. Recall that the complement mapping cm on Σ_{DNA} is defined by $cm(\mathtt{A}) = \mathtt{T}$ and $cm(\mathtt{C}) = \mathtt{G}$. By construction, we have $\mathrm{rc}(h(a_x)) = h(cm(a_x))$ for every $a_x \in A$. Consequently, $\mathrm{rc}(h(s)) = h(\mathrm{rc}(s))$ holds for every string $s \in A^*$. Moreover, for any distinct symbols $a_x, a_y \in A$ with $x \neq y$, the strings $h(a_x)$ and $h(a_y)$ do not overlap. The only overlap of $h(a_x)$ with itself is the trivial one of its full length. Finally, note that $|h(a_x)| = 2(k+1-j)$ for all $a_x \in A$; hence, $|h(s)| = 2(k+1-j)\,|s|$ for every string $s \in A^*$.

Now consider the instance $R = \{h(s) \mid s \in S\}$ over Σ^*_{DNA}. Let r be an arbitrary approximate solution to the SCS-RC instance R, which can be represented by a permutation $(i_1, \ldots, i_m)$ of $\{1, \ldots, m\}$ as $r = \langle h(s'_{i_1}), \ldots, h(s'_{i_m})\rangle$. By the non-overlapping property of the encoding h, as shown above, that is, overlaps can occur only at the boundaries of the blocks $h(a_i)$, r can be transformed into an approximate solution $s = \langle s'_{i_1}, \ldots, s'_{i_m}\rangle$ to the original instance S, and we have $|r| = 2(k+1-j)\,|s|$. This implies that

$$\mathrm{OPT}(R) = 2(k+1-j)\,\mathrm{OPT}(S).$$

Moreover, let $\mathrm{ALG}(R)$ denote the length of a β-approximate solution to the instance R. By the above argument, this solution can be transformed into a solution to the original instance S of length $\mathrm{ALG}(R)/2(k+1-j)$. Therefore, we obtain

$$\frac{\mathrm{ALG}(R)/2(k+1-j)}{\mathrm{OPT}(S)} = \frac{\mathrm{ALG}(R)}{\mathrm{OPT}(R)} \leq \beta,$$

which shows that a β-approximate solution for R yields a β-approximate solution for S.

6 Conclusion and Future Work

In this paper, we established an $\frac{8}{3}$-approximation algorithm for the SCS-RC problem and proved several hardness results. Our algorithm is based on computing an optimal constrained cycle cover on the distance graph, which can be viewed as a special case of the assignment-based relaxation for the generalized traveling salesman problem (GTSP), where the requirement of forming a single tour is relaxed to allow multiple disjoint cycles.

It is well known that the assignment-based relaxation of the classical traveling salesman problem reduces to finding a minimum-weight cycle cover in the standard sense, which is solvable in polynomial time. In contrast, the assignment-based relaxation of the GTSP is NP-hard, as shown in [13], where the hardness follows from a reduction to three-dimensional matching and involves clusters of size $\Theta(|V|^{\frac{2}{3}})$.

Our result demonstrates that the problem studied in this paper constitutes a nontrivial special case of the GTSP assignment-based relaxation that nevertheless admits a polynomial-time optimal algorithm. Identifying more precise structural conditions that separate tractable cases from intractable ones remains an important direction for future work.

Another important direction for future work is to further improve the approximation guarantee. One possible approach is to improve the compression ratio of the compression subroutine used in greedy-based algorithms [16]. Here, the compression ratio is defined as the ratio between the reduction from the total input length achieved by an approximate solution and that achieved by an optimal solution. For the standard SCS problem, approximation algorithms for the maximum asymmetric traveling salesman problem can be employed as black-box subroutines. In contrast, for the SCS-RC problem, one must additionally account for clusters formed by reverse-complement pairs, which constitutes a nontrivial extension of this approach. A deeper understanding of how such small clusters affect both computational complexity and approximability may lead to stronger approximation algorithms.

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, July 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, August 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 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.

7 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.

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

9 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.

10 Marek Karpinski and Richard Schmied. Improved inapproximability results for the shortest superstring and related problems. In *Proceedings of the Nineteenth Computing: The Australasian Theory Symposium - Volume 141*, CATS '13, pages 27–36, AUS, 2013. Australian Computer Society, Inc. URL: https://dl.acm.org/doi/10.5555/2525519.2525523.

11 John D. Kececioglu. *Exact and approximation algorithms for DNA sequence reconstruction*. PhD thesis, The University of Arizona, 1991. URL: https://www.proquest.com/dissertations-theses/exact-approximation-algorithms-dna-sequence/docview/304014003/se-2.

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

13 Charles E. Noon. *The Generalized Traveling Salesman Problem*. PhD thesis, The University of Michigan, 1988. URL: https://www.proquest.com/dissertations-theses/generalized-traveling-salesman-problem/docview/303673942/se-2.

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.

15 Virginia Vassilevska. Explicit inapproximability bounds for the shortest superstring problem. In Joanna Jędrzejowicz and Andrzej Szepietowski, editors, *Mathematical Foundations of Computer Science 2005*, pages 793–800, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/11549345_68.

16 Ryosuke Yamano and Tetsuo Shibuya. Improved Approximation Ratios for the Shortest Common Superstring Problem with Reverse Complements. In Philip Bille and Nicola Prezza, editors, *37th Annual Symposium on Combinatorial Pattern Matching (CPM 2026)*, volume 369 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:11, Dagstuhl, Germany, 2026. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CPM.2026.15.