Explicit Inapproximability Bounds for the Shortest Superstring Problem

Virginia Vassilevska[⋆]

Carnegie Mellon University, Pittsburgh PA 15213, USA
virgi@cs.cmu.edu

Abstract. Given a set of strings $S = \{s_1, \ldots, s_n\}$, the *Shortest Superstring* problem asks for the shortest string s which contains each s_i as a substring. We consider two measures of success in this problem: the *length* measure, which is the length of s, and the *compression* measure, which is the difference between the sum of lengths of the s_i and the length of s. Both the length and the compression versions of the problem are known to be MAX-SNP-hard. The only explicit approximation ratio lower bounds are by Ott: 1.000057 for the length measure and 1.000089 for the compression measure. Using a natural construction we improve these lower bounds to 1.00082 for the length measure and 1.00093 for the compression measure. Our lower bounds hold even for instances in which the strings are over a binary alphabet and have equal lengths. In fact, we show a somewhat surprising result, that the Shortest Superstring problem (with respect to both measures) is as hard to approximate on instances over a binary alphabet, as it is over any alphabet.

1 Introduction

Given a set of strings over some alphabet, the Shortest Superstring problem asks for the shortest string over the same alphabet which contains each of the given strings as a substring. The problem was first shown to be NP-hard by Maier and Storer [7]. As an optimization problem, it has two optimization measures: the length of the resulting superstring, and the compression, which is the difference between the sum of lengths of the given strings and the length of the superstring. The Shortest Superstring problem was shown by Blum et al. [4] to be MAX-SNP-hard with respect to both measures (over an unbounded alphabet), which implies that unless $P = NP$, there exists some $\epsilon > 0$ for which it is hard to approximate the optimal superstring to within a factor better than $(1 + \epsilon)$. Ott gave explicit approximation ratio lower bounds (assuming $P \neq NP$) for Shortest Superstring instances over a binary alphabet [12]. These ratios (1.000057 for the length measure and 1.000089 for the compression measure) are far from the best known upper bounds for the problem: 2.5 for the length measure by Sweedyk [13] and 1.625 for the compression measure by Bläser [3].

Using a natural reduction, we show a relationship between the approximability of the Vertex Cover and Shortest Superstring problems. Given a constant

⋆ Supported by the NSF ALADDIN Center (NSF Grant No. CCR-0122581)

approximation ratio lower bound for a class of graphs for which the optimal vertex cover is linear in the number of edges, we can obtain an inapproximability constant for the Shortest Superstring problem on equal length strings. Berman and Karpinski [2] and Karpinski [8] gave a series of inapproximability results for Vertex Cover on bounded degree graphs. Here we use the result for graphs of degree at most 5 to get that, unless $P = NP$, the Shortest Superstring problem is not 1.00082-approximable with respect to the length measure, and not 1.00093-approximable with respect to the compression measure. Notice that these constants, although small, improve on Ott's result by an order of magnitude. Moreover, these results have potential for much improvement if different inapproximability results for Vertex Cover are used.

Most hardness results for the Shortest Superstring problem, except Ott's result, are for instances over an unbounded alphabet. In fact, Ott [12] stresses that their result is the first APX-hardness result for instances over a binary alphabet. Small size alphabet instances are of interest because of their immediate relation to DNA sequencing, where an alphabet of size 4 (A,T,G,C) is used. The hardness of Shortest Superstring over a binary alphabet does not imply, however, that Shortest Superstring is not easier to approximate on smaller alphabet instances than in general. In this paper we show that the problem on a binary alphabet is just as hard to approximate as the general case, *i.e.* if one can approximate Shortest Superstring over a binary alphabet by a factor α in polynomial time, then the problem over any (finite) alphabet can be approximated by a factor α in polynomial time.

2 Preliminaries

In this section we define some terminology we will need.

Definition 1. *Given an alphabet Σ, a* string *over Σ is an element of Σ^*. Given two strings $s = s_1 \dots s_m$ and $t = t_1 \dots t_k$ over Σ, s is said to be a* substring *of t, if $|s| \leq |t|$ and there exists a j: $0 \leq j \leq k - m$ so that for every i: $1 \leq i \leq m$, $s_i = t_{j+i}$. t is said to be a* superstring *of s iff s is a substring of t. s is said to* overlap to the right *with t if there exists a j: $0 \leq j \leq m - 1$ so that for every i: $1 \leq i \leq m - j$, $s_i = t_{j+i}$. Then t is said to* overlap to the left *with s.*

Now consider the following procedure, $Induced(\pi, S)$, which given a set of n strings, $S = \{s_1, \dots, s_n\}$, and a permutation $\pi \in S_n$, greedily builds a superstring of S:

$Induced(\pi, S)$:
  $s \leftarrow s_{\pi(1)}$
  for i from 2 to n
    $s \leftarrow$ string obtained by maximally overlapping s to the right with $s_{\pi(i)}$

Definition 2. *Let $\pi \in S_n$ and let S be a set of n strings over some alphabet. We say π* induces *a superstring s_π on S if $s_\pi = Induced(\pi, S)$. Define $ov(\pi, S)$*

to be the amount of overlap induced by π *on* S*,* i.e. $ov(\pi, S) = (\sum_{i=1}^{n} |s_i|) - |Induced(\pi, S)|$.

Intuitively, the superstring induced on S by π is obtained by sequentially overlapping the strings in the order given by π.

Definition 3. *Given an alphabet* Σ *and a set of strings* $S = \{s_1, \ldots, s_n\} \subset \Sigma^*$ *such that no string in* S *is a substring of another string in* S*, the Shortest Superstring problem asks for the shortest string* s *which is a superstring of every* $s_i \in S$*. In terms of optimization, the* length *measure minimizes* $|s|$*, and the* compression *measure maximizes* $(\sum_{i=1}^{n} |s_i|) - |s|$*. When the compression measure is used, the problem is often referred to as the* maximum compression problem*.*

Note that the shortest superstring is the shortest length superstring over all superstrings induced by a permutation from S_n on S. Since the Shortest Superstring problem is defined on finite strings, here we consider the alphabet for the superstring instance to consist solely of characters occurring in the strings. With this definition, the alphabet size is always bounded by the sum of the string lengths.

3 Binary Alphabet Shortest Superstring

Until now the size of the underlying alphabet has been assumed to make a difference in the approximability of the Shortest Superstring problem. This may be related to the fact that a related problem, Shortest Common Supersequence, seems to be easier on instances over a small alphabet. For example, Jiang and Li [6] give an algorithm for Shortest Common Supersequence with an approximation ratio directly related to the size of the alphabet. In the next theorem we show that in the case of the Shortest Superstring problem, the alphabet size does not affect the approximability of the problem.

Theorem 1. *Suppose the Shortest Superstring problem can be approximated by a factor* α *on instances over a binary alphabet (with respect to either measure). Then the Shortest Superstring problem can be approximated by a factor* α *on instances over any alphabet.*

Proof. Given an alphabet $A = \{a_1, \ldots, a_k\}$ of size k, associate with a_i the binary string $s_i = 0^i(01)^{(k+1-i)}1^i$. Notice that if $i \neq j$, s_i does not overlap with s_j, and that the only way s_i overlaps with itself is by its whole length. Furthermore, all s_i have the same length, $2(k+1)$.

Consider an instance $T = \{t_1, \ldots, t_n\}$ over A and the instance $T' = \{t'_1, \ldots, t'_n\}$ obtained by substituting s_i for a_i. As noted earlier, we take A to contain only the characters present in the strings of T, so $|A| = k \leq \sum_i |t_i|$.

For all permutations $\pi \in S_n$ let s_π and s'_π be the superstrings induced by π on T and T' respectively. Then $|s'_\pi| = 2(k+1)|s_\pi|$. In particular, $|s'_{opt}| =$

$2(k+1)|s_{opt}|$ for the optimal permutation *opt* (in terms of the length measure for T). And if $|s'_\pi| \le \alpha|s'_{opt}|$, we have

$$|s_\pi| = \frac{|s'_\pi|}{2(k+1)} \le \frac{\alpha|s'_{opt}|}{2(k+1)} = \alpha|s_{opt}|.$$

For all permutations $\pi \in S_n$, let $ov_\pi = ov(\pi, T)$ and $ov'_\pi = ov(\pi, T')$. Then $ov'_\pi = 2(k+1) \cdot ov_\pi$. As above we have $ov'_{opt} = 2(k+1) \cdot ov_{opt}$ for the optimal (now in terms of overlaps) permutation *opt*. For $ov'_\pi \le \alpha \cdot ov'_{opt}$,

$$ov_\pi = \frac{ov'_\pi}{2(k+1)} \le \frac{\alpha \cdot ov'_{opt}}{2(k+1)} = \alpha \cdot ov_{opt}.$$

Hence if we have an α-approximation algorithm for the Shortest Superstring problem on binary strings, then we can use it to get an α-approximation for Shortest Superstring instances over any alphabet. The running time of the algorithm is polynomial since k is at most linear in the length of the input and the transformation can be carried out in polynomial time. □

4 Approximation Ratio Lower Bounds

In this section we derive explicit approximation ratio lower bounds (assuming $P \neq NP$) for the Shortest Superstring problem restricted to instances with equal length strings. We do this by a reduction from Vertex Cover, and by using the following theorem of Berman and Karpinski [2]:

Theorem 2 ([2]). *For any $0 < \epsilon < \frac{1}{2}$ it is NP-hard to decide whether an instance of Vertex Cover with $140n$ nodes and maximum degree at most 5 has its optimum above $(73-\epsilon)n$ or below $(72+\epsilon)n$.*

Moreover we will need the following fact concerning the reduction:

Claim. The Vertex Cover instances in Theorem 2 have at most $286n$ edges.

Proof of Claim: The instances in the reduction used in the proof of the theorem above have at most $12n$ nodes of degree 5, and the rest of the nodes have degree at most 4. Hence the instances considered have at most $30n + 256n = 286n$ edges. □

We are now prepared to derive the inapproximability bounds.

Theorem 3. *For any $\epsilon > 0$, unless $P = NP$, Shortest Superstring on instances with equal length strings is not approximable in polynomial time within a factor of*

- $1.00082 - \epsilon$ *with respect to the length measure, and*
- $1.00093 - \epsilon$ *with respect to the compression measure.*

Proof. Suppose we are given an instance of Vertex Cover $G = (V, E)$ with $|E| = m$. Let our alphabet contain a letter a for each vertex $a \in V(G)$, and our strings be $abab$ and $baba$ for each edge $e = (a, b)$.

Suppose G has a vertex cover S of size k. Then, assign each edge to one of its end points which is in S. If $e = (a, b)$ was assigned to a, then overlap $abab$ (to the left) with $baba$ to obtain $ababa$. Otherwise overlap them in the opposite order to obtain $babab$.

For every $b \in S$, consider all edges (a_i, b) assigned to b. For each one of these edges we have a string of the form ba_iba_ib. By consecutively overlapping these strings by 1 letter, we obtain a string s_b. By concatenating the s_b strings for all $b \in S$ we obtain a string s.

Claim. The string s has length $4m + k$.

Proof of Claim: The sum of the lengths of the original strings is $8m$ and each two strings corresponding to the same edge are overlapped by 3 symbols. This gives a total of $5m$ for the strings of the form $babab$. If all of these were overlapped by one letter we would get a compression of $(m - 1)$ since there are m of these strings. However, since the vertex cover is of size k, there are k groups with no overlap between them. So the length of the superstring is actually longer by $(k - 1)$ symbols. We have $|s| = 5m - (m - 1) + (k - 1) = 4m + k$ □

Now suppose for some $k \geq 1$ we have a superstring of length $4m + k$. First we show that wlog we may assume for every $(a, b) \in E$ that either $abab$ is overlapped to the right with $baba$ or vice-versa.

Suppose some $abab$ and $baba$ are not overlapped with each other. We will construct a new superstring of length $\leq 4m + k$ such that they do overlap.

Consider the permutation π of the strings which induces the superstring. Wlog, $abab$ occurs before $baba$, in π. In the worst case, there is a string $ba'ba'$ right after $abab$ overlapping with $abab$ to the right, and there is a string $a''ba''b$ right before $baba$ overlapping with $baba$ to the left (where clearly $a' \neq a \neq a''$). We can break these two overlaps moving all strings between $abab$ and $baba$ to the end of the permutation (without breaking any other overlaps), and then overlapping $abab$ with $baba$, for an overall gain of 1 letter overlap. After doing this for all edges we get a superstring of no greater length in which for each $(a, b) \in E$ either $abab$ is overlapped to the left with $baba$ or vice-versa.

After this transformation, the superstring s' is a concatenation of strings of the form

$$a_s \prod_i (b_i a_s b_i a_s),$$

where $\prod$ stands for iterated concatenation.

For an edge $(a, b) \in E$, if $abab$ overlaps to the left with $baba$, then put a in the vertex cover S. S is clearly a vertex cover since we used a vertex for each edge. If s' consists of t strings of the above form, $S = \bigcup_s a_s$ and $|S|$ is at most t. The length of the superstring is

$$5m - (m - 1) + (t - 1) = 4m + t$$

by an argument similar to the one given earlier. Since the length of the superstring did not increase due to our manipulations,

$$4m + t \leq 4m + k,$$

which yields $t \leq k$.

Therefore G has a vertex cover of size k iff the string set has a superstring of size $4m + k$.

Now we prove the inapproximability bounds. By Theorem 2, we have that for any $0 < \epsilon < \frac{1}{2}$ it is NP-hard to decide whether an instance of Vertex Cover with $140n$ nodes and at most $286n$ edges has its optimum above $(73 - \epsilon)n$ or below $(72 + \epsilon)n$.

Hence for Shortest Superstring on $2m \leq 572n$ strings of length 4 it is NP-hard to distinguish whether there is a superstring of length below $4m+(72+\epsilon)n$ or above $4m + (73 - \epsilon)n$. So if Shortest Superstring can be approximated within an α factor, then

$$\alpha \geq \frac{4m + (73 - \epsilon)n}{4m + (72 + \epsilon)n}.$$

Taking limits on both sides we get

$$\alpha \geq \lim_{\epsilon \to 0} \frac{4m + (73 - \epsilon)n}{4m + (72 + \epsilon)n} = \frac{4m + 73n}{4m + 72n} = 1 + \frac{1}{4\frac{m}{n} + 72}$$

But $4\frac{m}{n} \leq 286 \times 4 = 1144$ and so

$$\alpha \geq 1 + \frac{1}{1216} \geq 1.00082$$

Therefore, for any $\epsilon > 0$, Shortest Superstring on instances with equal length strings cannot be approximated within a factor of $1.00082 - \epsilon$, with respect to the length measure, unless $P = NP$.

When the length of the superstring is $4m+k$, the compression is $8m-(4m+k) = 4m-k$. So for the maximum compression on the strings from our reduction it is NP-hard to decide whether the optimum compression is above $4m-(72+\epsilon)n$ or below $4m - (73 - \epsilon)n$. If the compression can be approximated by a factor β, then

$$\beta \geq \frac{4m - (72 + \epsilon)n}{4m - (73 - \epsilon)n}$$

Taking limits on both sides,

$$\beta \geq \lim_{\epsilon \to 0} \frac{4m - (72 + \epsilon)n}{4m - (73 - \epsilon)n} = \frac{4m - 72n}{4m - 73n} =$$

$$= 1 + \frac{1}{\frac{4m}{n} - 73} \geq 1 + \frac{1}{1071} \geq 1.00093$$

Hence for any $\epsilon > 0$, Shortest Superstring on instances with equal length strings cannot be approximated within a factor of $1.00093 - \epsilon$, with respect to the compression measure, unless $P = NP$. $\square$

Using Theorems 1 and 3 we also get

Corollary 1. *For any $\epsilon > 0$, unless $P = NP$, Shortest Superstring on instances with equal length* binary *strings is not approximable in polynomial time within a factor of*

- $1.00082 - \epsilon$ *with respect to the length measure, and*
- $1.00093 - \epsilon$ *with respect to the compression measure.*

5 Conclusion

We have derived explicit approximation ratio lower bounds for the Shortest Superstring problem, when restricted to instances with equal length strings. These bounds are far from the best known upper bounds for the Shortest Superstring problem. The reduction given in this paper presents a promising avenue for improving the lower bounds further, since any better bounds for a class of Vertex Cover instances with an optimum linear in the number of edges immediately improves our result. This of course would only give lower bounds for the restricted version of Shortest Superstring, which may be weaker than the best lower bounds for the general problem. It is an interesting question whether Shortest Superstring on equal length strings is easier in terms of approximation than the general Shortest Superstring problem.

We have also shown that the alphabet size does not affect the approximability of Shortest Superstring. It is an open problem whether a similar result can be obtained for the related Shortest Common Supersequence problem, which is also known to be MAX-SNP-hard over a binary alphabet [11]. Our reduction exploited a property of the Shortest Superstring problem which is not present in the Shortest Common Supersequence problem. Hence, if the alphabet size does not affect the approximability of Shortest Common Supersequence, then proving this may require very different ideas from ours.

Our result on the alphabet importance for Shortest Superstring is significant since the main application of the Shortest Superstring problem is in DNA sequencing where the alphabet has only 4 symbols. Until now it was assumed that because of this restriction the real-world applications of the problem may be much better approximable. In this paper we have refuted this hope. But we also shed light on a very natural relation between Shortest Superstring and Vertex Cover, a problem which has been well-studied. Moreover, we conjecture that the relation between the two problems is much tighter than our reduction indicates, and that if Vertex Cover is not 2-approximable, then neither is Shortest Superstring.

6 Acknowledgments

I would like to thank Ryan Williams, Maverick Woo and my advisor Guy Blelloch for numerous helpful discussions, Avrim Blum for suggesting the Shortest Superstring problem, Uriel Feige for several insightful observations, and the three anonymous reviewers for their comments.

References

1. C. Armen, C. Stein, Short Superstrings and the Structure of Overlapping Strings. J. Comput. Biol. **2** (2) (1995) 307–332
2. P. Berman, M. Karpinski, On Some Tighter Inapproximability Results (Extended Abstract). ICALP (1999) 200–209
3. M. Bläser, An 8/13–Approximation Algorithm for the Asymmetric Maximum TSP. SODA (2002) 64–73
4. A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis, Linear Approximation of Shortest Superstrings. STOC (1991) 328–336
5. D. Breslauer,T. Jiang and Z. Jiang, Rotations of Periodic Strings and Short Superstrings. J. Algorithms. **24** (2) (1997) 340–353
6. T. Jiang, M. Li, On the Approximation of Shortest Common Supersequences and Longest Common Subsequences. SIAM J. Comput. **24** (5) (1995) 1122–1139
7. D. Maier and J.A. Storer. A Note on the Complexity of the Superstring Problem. Princeton University Technical Report 233, Department of Electrical Engineering and Computer Science (1977)
8. M. Karpinski, Approximating Bounded Degree Instances of NP-hard Problems. Proc. 13th Symp. on Fundamentals of Computation Theory, LNCS 2138, Springer **10** (2001) 24–34
9. H. Kaplan, N. Shafrir, The Greedy Algorithm for Shortest Superstrings. Information Processing Letters **93** (2005) 13–17
10. M. Middendorf, More on the Complexity of Common Superstring and Supersequence Problems. Theoretical Computer Science **125** (2) (1994) 205–228
11. M. Middendorf, On Finding Various Minimal, Maximal, and Consistent Sequences over a Binary Alphabet. Theoretical Computer Science **145** (1995) 317–327.
12. S. Ott, Lower Bounds for Approximating Shortest Superstrings over an Alphabet of Size 2. WG (1999) 55–64
13. Z. Sweedyk, A $2\frac{1}{2}$–Approximation Algorithm for Shortest Superstring. SIAM J. Comput. **29** (3) (1999) 954–986
14. J. Tarhio and E. Ukkonen, A Greedy Approximation Algorithm for Constructing Shortest Common Superstrings. Theoretical Computer Science **57** (1988) 131–145
15. J.S. Turner, Approximation Algorithms for the Shortest Common Superstring Problem. Information and Computation **83** (1989) 1–20