Explicit Inapproximability Bounds for the Shortest Superstring Problem

Virginia Vassilevska

Carnegie Mellon University, Pittsburgh, USA

The Shortest Superstring Problem (SSP)

Given an alphabet Σ, and a set of strings $S = \{s_1, \ldots, s_n\}$ over Σ, find a shortest string s over Σ which contains every s_i as a substring.

The Shortest Superstring Problem (SSP)

Given an alphabet Σ, and a set of strings $S = \{s_1, \ldots, s_n\}$ over Σ, find a shortest string s over Σ which contains every s_i as a substring.

A substring of s is a string of consecutive characters in s.

The Shortest Superstring Problem (SSP)

Given an alphabet Σ, and a set of strings $S = \{s_1, \ldots, s_n\}$ over Σ, find a shortest string s over Σ which contains every s_i as a substring.

A substring of s is a string of consecutive characters in s.

AB and BCD are substrings of ABCDE.

The Shortest Superstring Problem (SSP)

Given an alphabet Σ, and a set of strings $S = \{s_1, \ldots, s_n\}$ over Σ, find a shortest string s over Σ which contains every s_i as a substring.

A substring of s is a string of consecutive characters in s.

AB and BCD are substrings of ABCDE. BE is not.

The Shortest Superstring Problem (SSP)

Given an alphabet Σ, and a set of strings $S = \{s_1, \ldots, s_n\}$ over Σ, find a shortest string s over Σ which contains every s_i as a substring.

A substring of s is a string of consecutive characters in s.

AB and BCD are substrings of ABCDE. BE is not.

The shortest superstring of $\{\texttt{OVER, VERY, DOVE}\}$ is DOVERY.

The Shortest Superstring Problem (SSP)

The Shortest Superstring Problem (SSP)

An overlap of string s_1 with string s_2 is a suffix of s_1 which is the same as the prefix of s_2 of the same length.

The Shortest Superstring Problem (SSP)

An overlap of string s_1 with string s_2 is a suffix of s_1 which is the same as the prefix of s_2 of the same length.

To overlap s_1 with s_2 maximally means to find the maximum overlap ov of s_1 with s_2, and to attach to the front of s_2 the prefix of s_1 before ov.

E.g. $\{abc, bcd\} \longrightarrow abcd$.

The Shortest Superstring Problem cont.

The Shortest Superstring Problem cont.

Two ways to measure the quality of a superstring s:

The Shortest Superstring Problem cont.

Two ways to measure the quality of a superstring s:

- Length - the number of symbols $|s|$ in the string

The Shortest Superstring Problem cont.

Two ways to measure the quality of a superstring s:

- Length - the number of symbols $|s|$ in the string
- Compression - $[\sum_i |s_i|] - |s|$

The Shortest Superstring Problem cont.

Two ways to measure the quality of a superstring s:

- Length - the number of symbols $|s|$ in the string
- Compression - $[\sum_i |s_i|] - |s|$

As a superstring of $\{$`OVER, VERY, DOVE`$\}$, `DOVERY` yields a compression of 6.

The Shortest Superstring Problem cont.

The Shortest Superstring Problem cont.

SSP is

The Shortest Superstring Problem cont.

SSP is

- NP-Hard (Maier and Storer),

The Shortest Superstring Problem cont.

SSP is

- NP-Hard (Maier and Storer),
- MAX-SNP-Hard (Blum *et al.*),

The Shortest Superstring Problem cont.

SSP is

- NP-Hard (Maier and Storer),
- MAX-SNP-Hard (Blum *et al.*),
- 2.5 – approximable in terms of the length measure (Sweedyk),

The Shortest Superstring Problem cont.

SSP is

- NP-Hard (Maier and Storer),
- MAX-SNP-Hard (Blum *et al.*),
- 2.5 – approximable in terms of the length measure (Sweedyk),
- $\frac{2}{3}$ – approximable in terms of the compression measure (Kaplan *et al.*),

The Shortest Superstring Problem cont.

SSP is

- NP-Hard (Maier and Storer),
- MAX-SNP-Hard (Blum *et al.*),
- 2.5 – approximable in terms of the length measure (Sweedyk),
- $\frac{2}{3}$ – approximable in terms of the compression measure (Kaplan *et al.*),
- for a binary alphabet, unless $\mathsf{P} = \mathsf{NP}$, not approximable within 1.000057 (length) and 1.000089 (compression) (Ott).

Our Main Result

Unless P $=$ NP, for any $\varepsilon > 0$, SSP on equal length binary strings cannot be approximated in poly time within a factor of

- $1.00082 - \varepsilon$, with respect to the length measure,
- $1.00093 - \varepsilon$, with respect to the compression measure.

Alphabet

Alphabet

Does it matter whether the alphabet is large or small?

Alphabet

Does it matter whether the alphabet is large or small?

Clearly, if $\Sigma = \{0\}$ the problem is very easy to solve.

Alphabet

Does it matter whether the alphabet is large or small?

Clearly, if $\Sigma = \{0\}$ the problem is very easy to solve.

Are smaller alphabet instances easier to solve, at least in terms of approximation?

Alphabet

Does it matter whether the alphabet is large or small?

Clearly, if $\Sigma = \{0\}$ the problem is very easy to solve.

Are smaller alphabet instances easier to solve, at least in terms of approximation?

We show that if you can approximate *binary* alphabet instances within a factor α in polytime then *general* SSP is α-approximable.

Binary Alphabet

Binary Alphabet

Suppose one can approximate binary alphabet instances in polytime within a factor α.

Binary Alphabet

Suppose one can approximate binary alphabet instances in polytime within a factor α.

Given an instance $S = \{s_1, \ldots, s_n\}$ of SSP on any alphabet Σ, go through the strings in S and in linear time collect the *finite* subalphabet $\Sigma' \subseteq \Sigma$ of letters participating in the given strings.

Binary Alphabet

Binary Alphabet

Let $\Sigma' = \{\sigma_1, \ldots, \sigma_m\}$.

Transform $\sigma_i \rightarrow 0^i(01)^{m+1-i}1^i$.

Binary Alphabet

Let $\Sigma' = \{\sigma_1, \ldots, \sigma_m\}$.

Transform $\sigma_i \longrightarrow 0^i(01)^{m+1-i}1^i$.

For example, for $m = 3$,

$$\Sigma' \rightarrow \{00101011, 00010111, 00001111\}.$$

Binary Alphabet

Let $\Sigma' = \{\sigma_1, \ldots, \sigma_m\}$.

Transform $\sigma_i \to 0^i(01)^{m+1-i}1^i$.

For example, for $m = 3$,

$$\Sigma' \to \{00101011, 00010111, 00001111\}.$$

Properties of this transformation:

Binary Alphabet

Let $\Sigma' = \{\sigma_1, \ldots, \sigma_m\}$.

Transform $\sigma_i \rightarrow 0^i(01)^{m+1-i}1^i$.

For example, for $m = 3$,

$$\Sigma' \rightarrow \{00101011, 00010111, 00001111\}.$$

Properties of this transformation:

- for $i \neq j$, σ_i does not overlap with σ_j,

Binary Alphabet

Let $\Sigma' = \{\sigma_1, \ldots, \sigma_m\}$.

Transform $\sigma_i \to 0^i(01)^{m+1-i}1^i$.

For example, for $m = 3$,

$$\Sigma' \to \{00101011, 00010111, 00001111\}.$$

Properties of this transformation:

- for $i \neq j$, σ_i does not overlap with σ_j,
- σ_i overlaps with itself only by its whole length,

Binary Alphabet

Let $\Sigma' = \{\sigma_1, \ldots, \sigma_m\}$.

Transform $\sigma_i \to 0^i(01)^{m+1-i}1^i$.

For example, for $m = 3$,

$$\Sigma' \to \{00101011, 00010111, 00001111\}.$$

Properties of this transformation:

- for $i \neq j$, σ_i does not overlap with σ_j,
- σ_i overlaps with itself only by its whole length,
- for every i, $|\sigma_i| = 2(m+1)$.

Binary Alphabet cont.

Binary Alphabet cont.

The transformed σ_i behave like single letters and the superstrings of the new instance correspond exactly to those of the old one, and their length is exactly $2(m+1)$ times larger.

Binary Alphabet cont.

The transformed σ_i behave like single letters and the superstrings of the new instance correspond exactly to those of the old one, and their length is exactly $2(m+1)$ times larger.

Hence an α approximation of the binary alphabet instance can immediately be converted to an α approximation for the old instance.

Binary Alphabet cont.

The transformed σ_i behave like single letters and the superstrings of the new instance correspond exactly to those of the old one, and their length is exactly $2(m+1)$ times larger.

Hence an α approximation of the binary alphabet instance can immediately be converted to an α approximation for the old instance.

Binary alphabet SSP is just as hard to approximate as general SSP.

Lower Bounds

Lower Bounds

Karpinski: For any $0 < \varepsilon < \frac{1}{2}$ 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 $(73 - \varepsilon)n$ or below $(72 + \varepsilon)n$.

Lower Bounds

Karpinski: For any $0 < \varepsilon < \frac{1}{2}$ 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 $(73 - \varepsilon)n$ or below $(72 + \varepsilon)n$.

The graph instances in the reduction used have at most $286n$ edges.

Lower Bounds

Karpinski: For any $0 < \varepsilon < \frac{1}{2}$ 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 $(73 - \varepsilon)n$ or below $(72 + \varepsilon)n$.

The graph instances in the reduction used have at most $286n$ edges.

We'll efficiently reduce from Vertex Cover on these graphs to SSP.

The Reduction

Given an instance of Vertex Cover $[G = (V, E), K]$ reduce to an SSP instance as follows:

1. $\Sigma = V$
2. S consists of $abab$ and $baba$ for all $(a, b) \in E$

How The Reduction Works

How The Reduction Works

Suppose G has a vertex cover C of size k and $|E| = m$.

How The Reduction Works

Suppose G has a vertex cover C of size k and $|E| = m$.

Assign each edge to one of its end points which is in C.

How The Reduction Works

Suppose G has a vertex cover C of size k and $|E| = m$.

Assign each edge to one of its end points which is in C.

If $e = (a, b)$ was assigned to a, then overlap $abab$ (to the left) with $baba$ to obtain $ababa$. [Otherwise obtain $babab$.]

How The Reduction Works

Suppose G has a vertex cover C of size k and $|E| = m$.

Assign each edge to one of its end points which is in C.

If $e = (a, b)$ was assigned to a, then overlap $abab$ (to the left) with $baba$ to obtain $ababa$. [Otherwise obtain $babab$.]

Let $c \in C$. The strings corresponding to edges assigned to c can all be overlapped by a letter to get something like

$$cacacbcbcdcdc \ldots c$$

.

How The Reduction Works

Suppose G has a vertex cover C of size k and $|E| = m$.

Assign each edge to one of its end points which is in C.

If $e = (a, b)$ was assigned to a, then overlap $abab$ (to the left) with $baba$ to obtain $ababa$. [Otherwise obtain $babab$.]

Let $c \in C$. The strings corresponding to edges assigned to c can all be overlapped by a letter to get something like

$$cacacbcbcdcdc \ldots c$$.

This gives a superstring of length $4m + k$.

How The Reduction Works

How The Reduction Works

Suppose the SSP instance we constructed has a superstring s of length at most $4m + k$.

How The Reduction Works

Suppose the SSP instance we constructed has a superstring s of length at most $4m + k$.

If some $abab$ and $baba$ were not overlapped with each other, wlog assume $abab$ occurs before $baba$ in s.

How The Reduction Works

Suppose the SSP instance we constructed has a superstring s of length at most $4m + k$.

If some $abab$ and $baba$ were not overlapped with each other, wlog assume $abab$ occurs before $baba$ in s.

In the worst case we have $\ldots ababa'ba' \ldots a''ba''baba \ldots$.

How The Reduction Works

Suppose the SSP instance we constructed has a superstring s of length at most $4m + k$.

If some $abab$ and $baba$ were not overlapped with each other, wlog assume $abab$ occurs before $baba$ in s.

In the worst case we have $\ldots ababa'ba' \ldots a''ba''baba \ldots$.

We can *gain one symbol* overlap by moving $ba'ba' \ldots a''ba''b$ to the end of s and overlapping $abab$ with $baba$!

How The Reduction Works

How The Reduction Works

Hence wlog assume that in s for every edge (a, b) either $abab$ is maximally overlapped with $baba$ or vice versa.

How The Reduction Works

Hence wlog assume that in s for every edge (a, b) either $abab$ is maximally overlapped with $baba$ or vice versa.

To obtain a vertex cover C of G, for each edge (a, b) if $abab$ comes before $baba$ in s, put a in C. Otherwise put b in C.

How The Reduction Works

Hence wlog assume that in s for every edge (a, b) either $abab$ is maximally overlapped with $baba$ or vice versa.

To obtain a vertex cover C of G, for each edge (a, b) if $abab$ comes before $baba$ in s, put a in C. Otherwise put b in C.

The strings of the form $ababa$ overlap by at most one symbol and this symbol is in C by construction.

How The Reduction Works

Hence wlog assume that in s for every edge (a, b) either $abab$ is maximally overlapped with $baba$ or vice versa.

To obtain a vertex cover C of G, for each edge (a, b) if $abab$ comes before $baba$ in s, put a in C. Otherwise put b in C.

The strings of the form $ababa$ overlap by at most one symbol and this symbol is in C by construction.

The shortest possible string that can be obtained by overlapping them is of length $4m + |C|$. Hence $|C| \leq k$.

Putting it all together

Putting it all together

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

Putting it all together

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

For any $0 < \varepsilon < \frac{1}{2}$ it is NP-hard to decide whether an instance of Vertex Cover with $140n$ nodes and at most $286n$ edges has its optimum above $(73 - \varepsilon)n$ or below $(72 + \varepsilon)n$.

Putting it all together

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

For any $0 < \varepsilon < \frac{1}{2}$ it is NP-hard to decide whether an instance of Vertex Cover with $140n$ nodes and at most $286n$ edges has its optimum above $(73 - \varepsilon)n$ or below $(72 + \varepsilon)n$.

Hence for SSP on $2m \leq 572n$ strings of length 4 it is NP-hard to distinguish whether there is a superstring of length below $4m + (72 + \varepsilon)n$ or above $4m + (73 - \varepsilon)n$.

Lower Bound for the Length Measure

Lower Bound for the Length Measure

If SSP can be approximated within α, then

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

Lower Bound for the Length Measure

If SSP can be approximated within α, then

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

Taking limits on both sides we get

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

Lower Bound for the Length Measure

If SSP can be approximated within α, then

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

Taking limits on both sides we get

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

But $4\frac{m}{n} \leq 286 \times 4 = 1144$ and so $\alpha \geq 1.00082$

Lower Bound for the Compression Measure

Lower Bound for the Compression Measure

The compression in our reduction is $4m - k$.

Lower Bound for the Compression Measure

The compression in our reduction is $4m - k$.

So for the compression measure it is NP-hard to decide whether the optimum compression is above $4m - (72 + \varepsilon)n$ or below $4m - (73 - \varepsilon)n$.

Lower Bound for the Compression Measure

The compression in our reduction is $4m - k$.

So for the compression measure it is NP-hard to decide whether the optimum compression is above $4m - (72 + \varepsilon)n$ or below $4m - (73 - \varepsilon)n$.

If the compression can be approximated by a factor β, then

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

Lower Bound for the Compression Measure

The compression in our reduction is $4m - k$.

So for the compression measure it is NP-hard to decide whether the optimum compression is above $4m - (72 + \varepsilon)n$ or below $4m - (73 - \varepsilon)n$.

If the compression can be approximated by a factor β, then

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

Taking limits on both sides,

$$\beta \geq \lim_{\varepsilon \to 0} \frac{4m - (72 + \varepsilon)n}{4m - (73 - \varepsilon)n} = 1 + \frac{1}{\frac{4m}{n} - 73} \geq 1.00093$$

Summary

Unless $\mathsf{P} = \mathsf{NP}$, for any $\varepsilon > 0$, SSP on equal length strings cannot be approximated in poly time within a factor of

- $1.00082 - \varepsilon$, with respect to the length measure,
- $1.00093 - \varepsilon$, with respect to the compression measure.

Open Questions

- Is SSP on *equal length* strings easier than the *general* SSP in terms of approximation?
- Is SSP more tightly related to Vertex Cover?
- Can we obtain better hardness results if we relax our assumptions from P $\neq$ NP to something like NP $\notin n^{polylog(n)}$?
- Can one obtain similar results for Shortest Common Supersequence?

Thank You!