A Tutorial on Shortest Superstring Approximation

Marcin Mucha

December 17, 2007

1 Introduction

This is a tutorial on approximation algorithms for the Shortest Superstring Problem (SSP). My intention when writing it was to provide the foundations for actually doing research on this topic. In Section 2 I cover the basic definitions and observations and introduce the "Greedy Conjecture". In Section 3 I describe the standard framework for approximating SSP based on a overlap-reducing transformation of the instance and then reducing the problem to asymmetric maxTSP. This culminates in the 2.5-approximation algorithm as given by Kaplan et al. [5]. Several intermediate results are ommited here (e.g. [1] or [4]), as well as the algorithm of Sweedyk [7], which achieves the same approximation ratio, but is much more complicated. I also think the approach described here is the most promising one in terms of further improvements. In Section 4, I recall the basic results on the performance of the greedy algorithm, including the recent analysis of Kaplan and Shafrir [6] yielding a bound of 3.5 on the approximation factor.

2 Basics

2.1 Definition of the problem

In the *Shortest Superstring Problem* (SSP), we are given a set $S = \{s_1, \ldots, s_n\}$ of string over a finite alphabet Σ, and we have to find a shortest string s that contains all the s_i as substrings. In other words, s is a shortest superstring of all the s_i. Without loss of generality we assume that no s_i is a substring of s_j for $i \neq j$.
This problem is **NP**-hard (see [2]), so assuming $\mathbf{P} \neq \mathbf{NP}$, the best one can hope for, in terms of polynomial algorithms, is approximation.

2.2 The greedy algorithm

Let the *overlap* $\mathrm{ov}(s,t)$ of two strings s, t be the longest string y such, that $s = xy$ and $t = yz$ for some non-empty x, z. Also let the *prefix* $\mathrm{pr}(s,t)$ of s w.r.t. t be the string x in the above definition. So, $s = \mathrm{pr}(s,t)\mathrm{ov}(s,t)$. Notice that $\mathrm{pr}(s,t)t$ is the shortest string containing s and t in that order. We call this string a *merge* of s and t.
The following natural approach to SSP, called the *Greedy Algorithm* works surprisingly well. We pick two strings s_i, s_j with largest overlap from S (breaking ties arbitrarily) and replace them with their merge. The algorithm stops when there is only one string left and, obviously, this string is a superstring of all the s_i.
How good is this algorithm? Considering the set of $\{ab^k, b^kc, b^{k+1}\}$, we can see that the Greedy Algorithm cannot have an approximation factor better than 2. However, resolving the following conjecture due to Blum et al. [2] is a long standing and very important open problem:

Conjecture 1 (The Greedy Conjecture)**.** *The Greedy Algorithm has approximation factor* 2*.*

Blum et al. showed that this factor is at most 4, and Kaplan and Shafrir recently improved this bound to 3.5.

Exercise 1. *In the hard example given above, the Greedy Algorithm has two choices for the first step. If it picks the right one, it can actually end up with the optimal solution. Can you give an example where the Greedy Algorithm has to go wrong? Can you give an example where the alphabet has only* 2 *symbols?*

2.3 Overlap graph, prefix graph and reductions to TSP

The following two directed graphs are very good models of how the strings in S overlap with each other. The *overlap graph* of S is a complete directed graph with S as the vertex set, and the edge (s_i, s_j) having length $|\mathrm{ov}(s_i, s_j)|$. The *prefix graph* (also called the *distance graph*) is defined similarly, only the edge (s_i, s_j) has length $|\mathrm{pr}(s_i, s_j)|$.
Let $\langle s_{i_1}, s_{i_2}, \ldots, s_{i_n} \rangle$ be the string $\mathrm{pr}(s_{i_1}, s_{i_2})\mathrm{pr}(s_{i_2}, s_{i_3}) \ldots \mathrm{pr}(s_{i_{n-1}}, s_{i_n})s_n$. Obviously, it's the shortest string containing $s_{i_1}, s_{i_2}, \ldots, s_{i_n}$ in that order. Notice that the optimal solution has the form $\langle s_{i_1}, s_{i_2}, \ldots, s_{i_n} \rangle$ for some ordering $s_{i_1}, s_{i_2}, \ldots, s_{i_n}$.
The length of $\langle s_{i_1}, s_{i_2}, \ldots, s_{i_n} \rangle$ is equal to (prove it):

$$|\mathrm{pr}(s_{i_1}, s_{i_2})| + |\mathrm{pr}(s_{i_2}, s_{i_3})| + \ldots + |\mathrm{pr}(s_{i_{n-1}}, s_{i_n})| + |\mathrm{pr}(s_{i_n}, s_{i_1})| + |\mathrm{ov}(s_{i_n}, s_{i_1})|,$$

which is the length of the cycle $s_{i_1}, s_{i_2}, \ldots, s_{i_n}$ in the prefix graph of S increased by $|\mathrm{ov}(s_{i_n}, s_{i_1})|$. Thus, the length of the shortest TSP tour in the prefix graph of S is a lower bound on OPT.

Exercise 2. *Give an approximation factor preserving reduction of the Shortest Superstring Problem to the directed (asymmetric) version of TSP, i.e. show that α-approximation algorithm for the TSP can be used to α-approximate SSP.*

The above considerations suggest that reduction to asymmetric TSP might be useful in approximating SSP. Unfortunately, the best known approximation algorithms for asymmetric TSP have factors of order $\log n$, so this approach is not very useful.
Let us look again at a generic solution $\langle s_{i_1}, s_{i_2}, \ldots, s_{i_n} \rangle$ and this time express it's length in terms of the overlap graph:

$$\sum_{j=1}^{n} |s_j| - \sum_{j=1}^{n-1} |\mathrm{ov}(s_{i_j}, s_{i_{j+1}})|.$$

The right term in the above expression (called the total overlap of $\langle s_{i_1}, s_{i_2}, \ldots, s_{i_n} \rangle$) is the length of the path $s_{i_1}, \ldots, s_{i_n}$ in the overlap graph, so the longest TSP path in the overlap graph corresponds to the optimal solution for SSP. Longest TSP path in a directed graph can be approximated within constant factor. Notice however, that

Exercise 3. *Using a constant factor approximation of the directed MAX-TSP-path problem in the above manner does not lead to a good approximation algorithm for SSP.*

The reason why this approach does not work is that the strings in S can have huge overlaps, so the optimal solution can have a very large total overlap compared to it's length. In that case, sacrificing even a small constant fraction of the overlap is leads to very bad solutions. We will show that this obstacle can be overcome and approximation algorithms for MAX-TSP-path can in fact be used to approximate SSP.

3 Approximating the Shortest Superstring Problem

In the previous section we noticed that the Shortest Superstring Problem can be reduced to the Asymmetric Maximum TSP Path problem (in the overlap graph). However, good approximation of the latter do not lead to good approximation of SSP. The problem here is that using this approach we approximate the total overlap of the solution and not the total length and the total overlap of the optimal solution can be huge compared to its length. We now show that this difficulty can be overcome by transforming the instance in a certain way.

3.1 Overlap reduction

Let S be a set of strings — an instance of the Shortest Superstring Problem. We are going to construct R, another set of strings, such that:

1. strings in R do not overlap too much,
2. every $s \in S$ has a superstring in R, so any superstring for R induces a superstring for S,
3. optimal solution OPT_R for R is not much worse than the optimal solution OPT_S for S.

Once we construct R, we can find a good solution for R via reduction to Asymmetric Maximum TSP path problem (which is a viable approach because of (1)), and this solution will also be a good solution for S because of (2) and (3).
How to construct the set R? Here is the standard approach.

1. find a minimum cycle cover $\mathcal{C}$ in the prefix graph,
2. for each cycle $C \in \mathcal{C}$, construct a representative $r(C)$ containing all strings in C as substrings; let $C = \{s_{i_1}, \ldots, s_{i_k}\}$, we can take $r(C) = \langle s_{i_1}, \ldots, s_{i_k} \rangle$,
3. let R be a set of all the representatives.

Remark 4. *The minimum cycle cover in a graph can be found in polynomial time by reducing the problem to the bipartite weighted matching problem. As we shall see in Section 4, a minimum cycle cover in the prefix graph can actually be found in a much easier way.*

The idea here is that strings that end up in the same cycle of $\mathcal{C}$ overlap a lot, so substituting them with a single representative should not increase the size of the optimal solution too much, but it should reduce the overlap of the instance.
Let us formalize this intuition. First of all we will show that OPT_R is not much larger than OPT_S.

Lemma 5. *The total length $w(\mathcal{C})$ of all the cycles in the minimum cycle cover $\mathcal{C}$ is $\leq OPT$.*

Proof. This is because the total length of the minimum cycle cover in the prefix graph is not greater than the length of the minimum TSP tour in this graph, which is a lower bound for OPT as we noted in the previous section. □

Lemma 6. $OPT_R \leq 2OPT_S$.

Proof. Recall that every $r \in R$ is of the form $r = \langle s_{i_1}, \ldots, s_{i_k} \rangle$. Consider a slightly longer $\hat{r} = \langle s_{i_1}, \ldots, s_{i_k}, s_{i_1} \rangle$ and let $\hat{R}$ consists of these longer versions of representatives. Then clearly $\mathrm{OPT}_R \leq \mathrm{OPT}_{\hat{R}}$. We will show that $\mathrm{OPT}_{\hat{R}} \leq 2\mathrm{OPT}_S$.

Each string $\hat{r} \in \hat{R}$ begins and ends with the same string, call it $s(\hat{r})$. Let $S(\hat{\hat{R}})$ be the collection of all these $s(\hat{r})$-s. Clearly $\mathrm{OPT}_{S(\hat{R})} \leq \mathrm{OPT}(S)$. But notice that the optimal solution for $S(\hat{R})$ can be transformed into a solution for $\hat{R}$. Simply substitute each $s(\hat{r})$ with a corresponding r. Substitution of $s(r)$ by r increases the size of the solution by the length of the cycle C corresponding to r, so all the substitutions increase the size of the solution by $w(\mathcal{C})$ which is $\leq \mathrm{OPT}$ by Lemma 5. □

So, our transformation at most doubles the size of the optimal solution.
Now, let us prove that the elements of R do not overlap too much.

Lemma 7. *Let c_1, c_2 be two cycles in the minimum cycle cover $\mathcal{C}$ of the prefix graph and let $r(c_1)$ and $r(c_2)$ be the representatives for these cycles, as defined above. Then*

$$ov(r(c_1), r(c_2)) < w(c_1) + w(c_2).$$

Before we prove this Lemma we need a couple of technical definitions and observations.
For any cycle $c = s_{i_1}, \ldots, s_{i_k} \in \mathcal{C}$ in the prefix graph of S, let

$$s(c) = \mathrm{pr}(s_{i_1}, s_{i_2})\mathrm{pr}(s_{i_2}, s_{i_3}) \ldots \mathrm{pr}(s_{i_{k-1}}, s_{i_k})\mathrm{pr}(s_{i_k}, s_{i_1})$$

(we read out all the prefixes as we go along the cycle). So, for example, $r(c) = s(c)\mathrm{ov}(s_{i_k}, s_{i_1})$.
Let s^∞ be a semi-infinite string of the form $sss\ldots$.

Lemma 8. *For any cycle $c = s_{i_1}, \ldots, s_{i_k}$ in the prefix graph of S, every s_{i_j} is a substring of $[s(c)]^\infty$. Also, $r(c) = \langle s_{i_1}, \ldots, s_{i_k} \rangle$ is a substring of $[s(c)]^\infty$.*

Proof. Recall that for any s, t we have $s = \mathrm{pr}(s,t)\mathrm{ov}(s,t)$ so s is a substring (a prefix, in fact) of $\mathrm{pr}(s,t)t$.
Thus, s_{i_j} is a prefix of $\mathrm{pr}(s_{i_j}, s_{i_{j+1}})s_{i_{j+1}}$, which in turn is a prefix of $\mathrm{pr}(s_{i_j}, s_{i_{j+1}})\mathrm{pr}(s_{i_{j+1}}, s_{i_{j+2}})s_{i_{j+2}}$, and so on. By continuing in this manner along the cycle c we get the claim for s_{i_j}. The same reasoning works for $r(c)$. □

Interestingly, "the reverse" is also true:

Lemma 9. *If all strings in $\hat{S} \subseteq S$ are substrings of t^∞, then there exists a cycle c of length $|t|$ in the prefix graph of S, that contains all these strings.*

Proof. If a string s is a substring of t^∞, then it appears in t^∞ every $|t|$ characters. This defines a circular ordering of the strings in $\hat{S}$ and it is easy to see that this ordering gives a cycle of length $|t|$ in the prefix graph. □

Proof (of Lemma 7). Let x, $|x| \geq w(c_1) + w(c_2)$ be the overlap of $r(c_1)$ and $r(c_2)$. Then x is a substring of $[s(c_1)]^\infty$ (because $r(c_1)$ is) and of $[s(c_2)]^\infty$ (because $r(c_2)$ is). Let x_1 be a prefix of x of length $w(c_1)$ and let x_2 be a prefix of x of length $w(c_2)$. Obviously, x is a prefix of x_1^∞ and of x_2^∞.
Now, since $|x| >= w(c_1) + w(c_2) = |x_1| + |x_2|$, we have $x_1x_2 = x_2x_1$. But this means that $x_1^k x_2^k = x_2^k x_1^k$ for any k (by induction), so $x_1^\infty = x_2^\infty$.
Any string in c_1 is a substring of x_1^∞ (by Lemma 8, since x_1 is a cyclic rotation of $s(c_1)$), so it's also a substring of x_2^∞. Also, any substring in c_2 is a substring of x_2^∞. So, by Lemma 9, all the strings in cycles c_1 and c_2 are contained in a single cycle of length $w(c_2)$ which contradicts the assumption that c_1 and c_2 are cycles in the minimum cycle cover. □

An immediate consequence of Lemma 7 and Lemma 5 is the following

Lemma 10. *The total overlap of any solution for the set R of representatives is $\leq 2OPT_S$.*

3.2 Reducing Shortest Superstring to maxTSP

In this section we show how the reduction defined in the previous section can be used to reduce SSP to directed Max TSP.
First, consider the following algorithm.

1. For a given set of strings S, construct a set of representatives R;
2. Return the concatenation of all the strings in R.

Quite surprisingly

Theorem 11. *The above algorithm is a 4-approximation of SSP.*

Proof. Consider the optimal solution for the set R. The length OPT_R of this solution is $\leq 2\mathrm{OPT}_S$. Also the total overlap of this optimal solution $\leq 2\mathrm{OPT}_S$ by Lemma 10. But the total length $|R|$ of the strings in R is exactly equal to the sum of length OPT_R of the optimal solution and its total overlap, so $|R| \leq 4\mathrm{OPT}_S$. □

To achieve better approximation we need to use directed Max TSP Path approximation to find a solution with good overlap for strings in R.

Theorem 12. *Given an α-approximation for directed Max-TSP Path, we can get a $(4-2\alpha)$-approximation for SSP.*

Proof. The approach here is obvious. We first construct the set R of representatives and then use TSP approximation to find a solution for R achieving an overlap of α times the optimum overlap. The length of this solution is

$$|R| - \alpha \mathrm{ov}(R) = |R| - \mathrm{ov}(R) + (1-\alpha)\mathrm{ov}(R) = \mathrm{OPT}_R + (1-\alpha)\mathrm{ov}(R)$$

(here, $\mathrm{ov}(R)$ denotes the total overlap of the optimal solution for R). Since $\mathrm{OPT}_R \leq 2\mathrm{OPT}_S$ and $\mathrm{ov}(R) \leq 2\mathrm{OPT}_S$, the length of this solution is $\leq (4-2\alpha)\mathrm{OPT}_S$. □

Concatenation of the strings in R corresponds to 0-approximation in the above theorem, and so it gives 4-approximation for SSP, as we already know. If we use $\frac{1}{2}$-approximation based on maximum cycle covers, we get 3-approximation for SSP. Finally, using the $\frac{2}{3}$-approximation of Kaplan et al. [5] (the best known result), we get $2\frac{2}{3}$-approximation for SSP.

3.3 Better overlap reduction

Since the best known Max TSP approximation algorithm has a factor of $\frac{2}{3}$ and it does not look like a better algorithm is coming any time soon[1], we need to look for other ways of improving the SSP approximation. One possible direction is improving Lemma 10, i.e. the bound on the total overlap of the optimal solution for R. This kind of result has been published by Breslauer et al. [3] and can be used in conjunction with results of Kaplan et al. [5] to achieve the best known approximation factor of 2.5 for SSP[2].
Let us first introduce a couple of definitions.
A string x is *a factor* of s if $s = x^i y$ for $i > 0$ and y a prefix of x (y may be empty). If k is the length of a factor of s, k is called *a period* of s. *THE factor* of s is the shortest factor of s, denoted $\mathrm{factor}(s)$. The length of the factor of s is called *THE period* of s and is denoted $\mathrm{per}(s)$. In case of semi-infinite strings (i.e. infinite sequences of letters from the alphabet), we call such a string s periodic if $s = xs$ for some non-empty x, the shortest such x is called the factor of s, and it's length is called the period of s. Two strings s, t (finite or not) are called *equivalent* if their factors are cyclic shifts of each other.
Notice that

Lemma 13. *Let c be a cycle in a minimum cycle cover of the prefix graph of S and let $r(c)$ be the representative for c. Then per$(r(c)) = w(c)$.*

Proof. First of all $w(c)$ is obviously a period of $r(c)$ because $r(c)$ is a substring of

$$[\mathrm{pr}(s_1, s_2)\mathrm{pr}(s_2, s_3)\ldots\mathrm{pr}(s_{k-1}, s_k)\mathrm{pr}(s_k, s_1)]^\infty,$$

where $c = s_1 s_2 \ldots s_k$. Why is it THE period of $r(c)$, i.e. why there is no smaller period? Suppose that $r(c)$ has a factor f shorter than $w(c)$. Then $r(c)$ is a substring of f^∞, so all of the s_i are also substrings of f^∞ (because they are substrings of $r(c)$) and then by Lemma 9 there is a cycle shorter than c containing all of s_i. This is a contradiction since we assumed that c was a cycle in the minimum cycle cover of the prefix graph of S. □

Breslauer et al. [3] prove the following

[1] The factor of $\frac{2}{3}$ seems optimal if you use maximum directed cycle cover with no 2-cycles. It's NP-hard to find these, but Kaplan et al. [5] manage to achieve $\frac{2}{3}$-approximation anyway — a really impressive result. Beating this bound is probably going to be very hard.

[2] The same factor was earlier achieved by Sweedyk [7] using a different, much more complicated approach

Lemma 14. *Let α be a periodic semi-infinite string. There exists an integer $p < per(s)$, such that for any (finite) string s inequivalent to α*

$$ov(s, \alpha[p, \infty]) < per(s) + \frac{1}{2}per(\alpha).$$

Also, if $per(s) \leq per(\alpha)$

$$ov(s, \alpha[p, \infty]) < \frac{2}{3}(per(s) + per(\alpha)).$$

Moreover, such p can be found in time $O(per(\alpha))$.

For a (slightly technical) proof the reader is referred to [3]. Let us consider what the above lemma is saying. From the classic periodicity lemma it follows that for any strings s and α as above

$$\mathrm{ov}(s, \alpha) < \mathrm{per}(s) + \mathrm{per}(\alpha).$$

This (together with 13) can in fact be used to prove Lemma 7. Now, Lemma 14 says, that we can modify α a bit by cutting off a short prefix in such a way, that maximum possible overlap with a nonequivalent string becomes much shorter. The way to use this Lemma for SSP approximation is to find representatives that have even smaller overlap than before.

Remark 15. *It is clear that if we can give a better bound on the total overlap of representatives, we can also give a better bound on the approximation factor (see proof of Theorem 12. From this point of view, it seems reasonable to select representatives that have small overlap. Overall, however, what we are doing here seems absurd. After all, we ARE looking for the shortest superstring, so it IS in our best interest to make the representatives overlap a lot. We will resolve this apparent contradiction later in this section.*

The following is a rephrasing of a result due to Breslauer et al:

Theorem 16. *It is possible to choose representatives $\tilde{R}$ for cycles in the minimum cycle cover of the prefix graph of S in such a way, that $OPT_{\tilde{R}} \leq 2OPT_S$ and for any two cycles c_1, c_2*

$$ov(r(c_1), r(c_2)) < w(c_1) + \frac{1}{2}w(c_2),$$

and also

$$ov(r(c_1), r(c_2)) < \frac{2}{3}(w(c_1) + w(c_2))$$

if $w(c_1) \leq w(c_2)$.

Remark 17. *The second inequality in the claim of the above Theorem will not be used in our algorithms. We give it here, because investigating the possibility of using this inequality seems to be a very promising direction of research.*

Proof. Consider any cycle $c = s_1, \ldots, s_k$ in the minimum cycle cover, let

$$\alpha = \mathrm{pr}(s_1, s_2)\mathrm{pr}(s_2, s_3) \ldots \mathrm{pr}(s_{k-1}, s_k)\mathrm{pr}(s_k, s_1)$$

and consider the semi-infinite string α^∞. Let p be as in Lemma 14. W.l.o.g. assume that position p actually lies in the $\mathrm{pr}(s_1, s_2)$ part of α (if not we can always start enumerating the vertices of c in a different vertex).
Now, in our previous approach we could take $r(c) = \langle s_2, s_3, \ldots, s_k, s_1\rangle$ as a representative for c. We could also take $\hat{r}(c) = \langle s_1, s_2, s_3, \ldots, s_k, s_1\rangle$ as a representative and still get $\mathrm{OPT}_R \leq 2\mathrm{OPT}_S$ even though this string is unnecessarily long (see the proof of Lemma 6 and string $\hat{r}$ in this proof). Now the idea is clear. We take $\tilde{r}(c)$ to be $\mathrm{pr}(s_1, s_2)[p..]\langle s_2, s_3, \ldots, s_k, s_1\rangle$. This $\tilde{r}(c)$ is a suffix of $\hat{r}(c)$. Also, $r(c)$ is a suffix of $\tilde{r}(c)$. So for a collection $\tilde{R}$ of representatives defined in this manner we get $\mathrm{OPT}_{\tilde{R}} \leq 2\mathrm{OPT}_S$.
Obviously, we also get the overlap inequalities claimed in the theorem. □

Lemma 18. *The total overlap in the optimum solution for $\tilde{R}$ defined above is $\leq 1.5 OPT_S$.*

Proof. Consider the optimum solution and use the inequality $\mathrm{ov}(r(c_1), r(c_2)) < w(c_1) + \frac{1}{2}w(c_2)$ for every pair of neighbouring representatives. □

The following is an improvement on Theorem 12

Theorem 19. *Given an α-approximation for directed Max-TSP Path, we can get a $(3.5 - 1.5\alpha)$-approximation for SSP.*

Proof. Identical to the proof of Theorem 12. □

By plugging in $\frac{2}{3}$-approximation for Max-TSP-Path due to Kaplan et al. [5] we get

Corollary 20. *It is possible to approximate SSP with factor* 2.5.

Recall Remark 15. What we have just done is we picked representatives $\tilde{R}$, slightly longer than the previous ones (R) and with much smaller overlap and because of that, we were able to achieve a better approximation ratio. This is counterintuitive, because the representatives R should actually produce a better solution. Luckily we can prove the following:

Theorem 21. *Using the original representatives R yields a $(3.5 - 1.5\alpha)$-approximation for SSP, where α is the approximation factor for the directed Max-TSP-Path approximation used. In fact, any set of representatives with total length $\leq |\tilde{R}|$ is enough.*

Proof. Consider the sets R and $\tilde{R}$ of representatives. When selecting the representative $r(c)$ for R use the cyclic rotation of the cycle c which yields the shortest $r(c)$. This guarantees that $|R| \leq |\tilde{R}|$. Let ov_R and $\mathrm{ov}_{\tilde{R}}$ be the total overlaps in optimal solutions for R and $\tilde{R}$ respectively. The total length of the approximated solution for R is

$$\leq |R| - \alpha \mathrm{ov}_R \leq |R| - \alpha(|R| - \mathrm{OPT}_R) \leq (1-\alpha)|R| + \alpha \mathrm{OPT}_R \leq (1-\alpha)|\tilde{R}| + 2\alpha \mathrm{OPT}.$$

Because of Lemma 18 we have $|\tilde{R}| \leq 3.5\mathrm{OPT}$, so the above expression is

$$\leq (1-\alpha)3.5\mathrm{OPT} + 2\alpha\mathrm{OPT} = (3.5 - 1.5\alpha)\mathrm{OPT},$$

as required. □

So the tricks in Breslauer's theorem are necessary in bounding of our algorithm's approximation factor but not in the algorithm itself (which Breslauer et al. do not seem to be aware of).

4 Analysis of the Greedy Algorithm

TODO

References

[1] Chris Armen and Clifford Stein. A 2 2/3-approximation algorithm for the shortest superstring problem. In *CPM*, pages 87–101, 1996.

[2] Avrim Blum, Tao Jiang, Ming Li, John Tromp, and Mihalis Yannakakis. Linear approximation of shortest superstrings. pages 328–336, 1991.

[3] Dany Breslauer, Tao Jiang, and Zhigen Jiang. Rotations of periodic strings and short superstrings. *J. Algorithms*, 24(2):340–353, 1997.

[4] Artur Czumaj, Leszek Gasieniec, Marek Piotrow, and Wojciech Rytter. Parallel and sequential approximations of shortest superstrings. In *Scandinavian Workshop on Algorithm Theory*, pages 95–106, 1994.

[5] 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.

[6] Haim Kaplan and Nira Shafrir. The greedy algorithm for shortest superstrings. *Inf. Process. Lett.*, 93(1):13–17, 2005.

[7] Z. Sweedyk. A $2\frac{1}{2}$-approximation algorithm for shortest superstring. *SIAM J. Comput.*, 29(3):954–986, 1999.