Approximating Shortest Superstring Problem Using de Bruijn Graphs

Alexander Golovnev[1], Alexander S. Kulikov[2,3], and Ivan Mihajlin[4]

[1] New York University
[2] St. Petersburg Department of Steklov Institute of Mathematics
[3] Algorithmic Biology Laboratory, St. Petersburg Academic University
[4] St. Petersburg Academic University

Abstract. The best known approximation ratio for the shortest superstring problem is $2\frac{11}{23}$ (Mucha, 2012). In this note, we improve this bound for the case when the length of all input strings is equal to r, for $r \leq 7$. E.g., for strings of length 3 we get a $1\frac{1}{3}$-approximation. An advantage of the algorithm is that it is extremely simple both to implement and to analyze. Another advantage is that it is based on de Bruijn graphs. Such graphs are widely used in genome assembly (one of the most important practical applications of the shortest common superstring problem). At the same time these graphs have only a few applications in theoretical investigations of the shortest superstring problem.

1 Introduction

1.1 Problem Statement

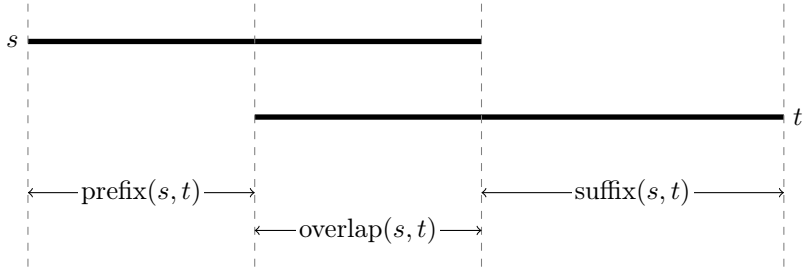
The *superstring problem* (also known as shortest common superstring problem, SCS, or shortest superstring problem, SSP) is: given n strings $s_1, \ldots, s_n$ to find a shortest string containing each s_i as a substring. By *r-superstring problem* (or just r-SCS) we denote the SCS problem for the special case when all input strings have length exactly r. Gallant et al. [10] showed that both SCS over the binary alphabet and 3-SCS are NP-hard, while 2-SCS can be solved in linear time. Crochemore et al. [7] proved that 2-SCS with multiplicities can be solved in quadratic time. (Note however that when both parameters, the length of input strings and the size of the alphabet, are bounded by constants then the problem degenerates since then the number of possible input strings is bounded by a constant.) The problem has received a lot of attention as it is interesting as a purely theoretical problem and has many practical applications including genome assembly and data compression.

In this note, we present a simple polynomial time algorithm that finds an $(r^2+r-4)/(4r-6)$-approximation to r-SCS. This is better than the best known approximation ratio $2\frac{11}{23}$ by Mucha [16] for $r = 3, \ldots, 7$. The algorithm first finds an approximate longest traveling salesman path in the overlap graph. It then finds an approximate shortest rural postman path in the de Bruijn graph. We show that if a permutation of the input strings given by one of these two paths does not give a good enough superstring then the other permutation does.

1.2 General Setting

For strings s and t by *overlap*(s,t) we denote the longest suffix of s that is also a prefix of t. By *prefix*(s,t) we denote the first $|s| - |\text{overlap}(s,t)|$ symbols of s. Similarly, *suffix*(s,t) is the last $|t| - |\text{overlap}(s,t)|$ symbols of t. Clearly, for any strings s and t,

$$\text{prefix}(s,t) \circ \text{overlap}(s,t) = s, \text{overlap}(s,t) \circ \text{suffix}(s,t) = t\,.$$



E.g.,

$$\text{overlap}(\texttt{ABACBA}, \texttt{BABCA}) = \texttt{BA},\ \text{prefix}(\texttt{ABACBA}, \texttt{BABCA}) = \texttt{ABAC}\,.$$

For a non-empty string s, by *prefix*(s) and *suffix*(s) we denote the string resulting from s by removing the last and the first symbol, respectively.

Now let $\mathcal{S} = \{s_1, \ldots, s_n\}$ be a set of strings over an alphabet Σ and s be a superstring of $\mathcal{S}$. By $\text{OPT}(\mathcal{S})$ we denote the length of a shortest possible superstring for $\mathcal{S}$. A *compression* of s (w.r.t. $\mathcal{S}$) is

$$|s_1| + |s_2| + \cdots + |s_n| - |s|\,.$$

Clearly minimizing the length of a superstring corresponds to maximizing the compression.

2 Known Results for SCS and Related Graph Problems

2.1 Superstring Problem

Improving the approximation ratio for the SCS problem is interesting both from practical and theoretical points of view. In this subsection, we review known results in this direction. Table 1 shows the sequence of known approximation algorithms and inapproximability results (under the P≠NP assumption) both for minimizing the length and maximizing the compression of superstrings. The well-known Greedy Conjecture [4] says that repeatedly combining two strings with maximal overlap gives a 2-approximation for SCS. Blum et al. [4] proved that this simple algorithm has ratio 4, and Kaplan and Shafrir [13] improved the ratio to 3.5.

Vassilevska [25] showed that an α-approximation of SCS over the binary alphabet implies an α-approximation over any alphabet. Hence, approximating SCS over the binary alphabet cannot be easier than for an arbitrary alphabet.

ratio	authors	year
	approximating SCS	
3	Blum, Jiang, Li, Tromp and Yannakakis [4]	1991
$2\frac{8}{9}$	Teng, Yao [23]	1993
$2\frac{5}{6}$	Czumaj, Gasieniec, Piotrow, Rytter [8]	1994
$2\frac{50}{63}$	Kosaraju, Park, Stein [15]	1994
$2\frac{3}{4}$	Armen, Stein [1]	1994
$2\frac{50}{69}$	Armen, Stein [2]	1995
$2\frac{2}{3}$	Armen, Stein [3]	1996
$2\frac{25}{42}$	Breslauer, Jiang, Jiang [5]	1997
$2\frac{1}{2}$	Sweedyk [21]	1999
$2\frac{1}{2}$	Kaplan, Lewenstein, Shafrir, Sviridenko [12]	2005
$2\frac{1}{2}$	Paluch, Elbassioni, van Zuylen [18]	2012
$2\frac{11}{23}$	Mucha [16]	2013
	approximating compression	
$\frac{1}{2}$	Tarhio, Ukkonen [22]	1988
$\frac{1}{2}$	Turner [24]	1989
$\frac{2}{3}$	Kaplan, Lewenstein, Shafrir, Sviridenko [12]	2005
$\frac{2}{3}$	Paluch, Elbassioni, van Zuylen [18]	2012
	inapproximability for SCS	
$1\frac{1}{17245}$	Ott [17]	1999
$1\frac{1}{1216}$	Vassilevska [25]	2005
$1\frac{1}{332}$	Karpinski, Schmied [14]	2012
	inapproximability for compression	
$1\frac{1}{11216}$	Ott [17]	1999
$1\frac{1}{1071}$	Vassilevska [25]	2005
$1\frac{1}{203}$	Karpinski, Schmied [14]	2012

Table 1: Known approximation ratios and inapproximability results for length and compression of superstrings

Note that SCS is a typical *permutation problem*: if we know the order of the input strings in a shortest superstring then we can recover this superstring by overlapping the strings in this given order. For this reason, it will be convenient for us to identify a superstring with the order of input strings in it. Below we describe several related graph permutation problems.

2.2 Prefix/overlap Graphs and Traveling Salesman Problem

Many known approximation algorithms for SCS work with the so-called overlap graph. The *overlap graph* $\mathrm{OG}(\mathcal{S})$ of the set of strings $\mathcal{S} = \{s_1, \ldots, s_n\}$ is a complete weighted directed graph on a set of vertices $V = \{1, \ldots, n\}$. The weight of an edge from i to j equals $|\mathrm{overlap}(s_i, s_j)|$. It is easy to see that solving SCS corresponds to solving the *asymmetric maximum traveling salesman path* (MAX-ATSP) problem in $\mathrm{OG}(\mathcal{S})$ where one is asked to find a longest path visiting each vertex of the graph exactly once (such a path is called *Hamiltonian*). Note that the length of any Hamiltonian path in this graph equals the compression of the corresponding superstring. The best known approximation ratio 2/3 for MAX-ATSP is due to Kaplan et al. [12]. This immediately gives a 2/3-approximation for the compression. Also, Breslauer et al. [5] showed that an α-approximation for MAX-ATSP implies a $3.5 - 1.5\alpha$ approximation for SCS. Plugging in the result by Kaplan et al. [12] gives a 2.5-approximation for SCS.

An alternative way is to find a *minimum traveling salesman path* (MIN-ATSP) in the *prefix graph* $\mathrm{PG}(\mathcal{S})$ where vertices i and j are joined by an edge of weight $|\mathrm{prefix}(s_i, s_j)|$. However MIN-ATSP cannot be approximated within any polynomial time computable function unless P=NP [20].

2.3 De Bruijn Graphs and Rural Path Problem

Another important concept is the de Bruijn graph $\mathrm{DG}(\mathcal{S})$. In this graph each input string $s_i \in \mathcal{S}$ is represented as a directed (unweighted) edge from $\mathrm{prefix}(s_i)$ to $\mathrm{suffix}(s_i)$. De Bruijn graphs are widely used in genome assembly, one of the practical applications of the SCS problem [19]. A useful property of de Bruijn graphs is the following: if $\mathcal{S}$ is the set of all substrings of length k of some unknown string s (this is called a *k-spectrum of s*) then we can solve SCS for $\mathcal{S}$ in polynomial time. Indeed, in this case there is an Eulerian path in the de Bruijn graph $\mathrm{DG}(\mathcal{S})$ spelling the string s. The advantage is that an Eulerian path in a graph can be found in linear time (as opposed to Hamiltonian path that is NP-hard to find). The found Eulerian path in $\mathrm{DG}(\mathcal{S})$ does not necessarily need to spell the initial string s (as a graph may contain many Eulerian paths) but it spells a shortest superstring. See Figure 1 for an illustration. A more detailed description of this algorithm can be found, e.g., in [19].

In general, solving the r-SCS problem corresponds to finding a shortest rural postman path in the following *extended de Bruijn graph* $\mathrm{EDG}(\mathcal{S})$: the set of vertices is Σ^{r-1}, and every two vertices s and t are joined by a directed edge of weight $|\mathrm{suffix}(s, t)|$. A path $t_1, \ldots, t_k$ spells a string of length

$$|t_1| + |\mathrm{suffix}(t_1, t_2)| + |\mathrm{suffix}(t_2, t_3)| + \cdots + |\mathrm{suffix}(t_{k-1}, t_k)|\,.$$

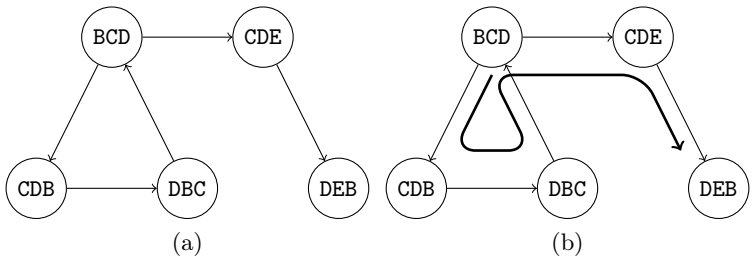


Fig. 1: Solving SCS for a k-spectrum of an unknown string s is easy. (a) De Bruijn graph of a set of strings {CDEB, CDBC, DBCD, BCDB, BCDE}. (b) An Eulerian path in this graph spells a shortest superstring BCDBCDEB.

Thus, the extended de Bruijn graph may be viewed as a weighted analogue of de Bruijn graph. The *shortest directed rural postman path* (DRPP) problem is: given a graph and a subset of edges to find a shortest path going through all these edges. DRPP has many practical applications (see, e.g., [9], [11]), and many papers study heuristic algorithms for it ([6], [9], [11]). At the same time almost no non-trivial theoretical bounds are known for DRPP.

This approach is particularly useful for solving 2-SCS. For this, we first construct the de Bruijn graph of the given set of 2-strings, then for each weakly connected component we add edges between imbalanced vertices (i.e., vertices with non-zero difference of in-degree and out-degree) so that the resulting component contains an Eulerian path. Finally, we add edges between components so that the graph contains an Eulerian path. Figure 2 gives an example. For a more detailed explanation of this algorithm see [10]. Crochemore et al. [7] used a similar technique to solve 2-SCS with multiplicities.

Note that the algorithm described above works for 2-SCS, but not for general r-SCS for the following reason: in case of 2-SCS, strings from different weakly connected components do not share letters (and hence have empty overlap) so the components can be traversed in any order.

3 Algorithm

In this section, we present a simple $(r^2+r-4)/(4r-6)$-approximation algorithm for the r-SCS problem. This ratio is better than the best known ratio $2\frac{11}{23}$ [16] for $r \leq 7$. Before presenting the algorithm we explain its main idea for the case of 3-strings.

3.1 Informally

Let $\mathcal{S} \subseteq \Sigma^3$ be a set of n strings of length 3. Note that $n+2 \leq \mathrm{OPT}(\mathcal{S}) \leq 3n$ (the former inequality corresponds to the case when in a shortest superstring all

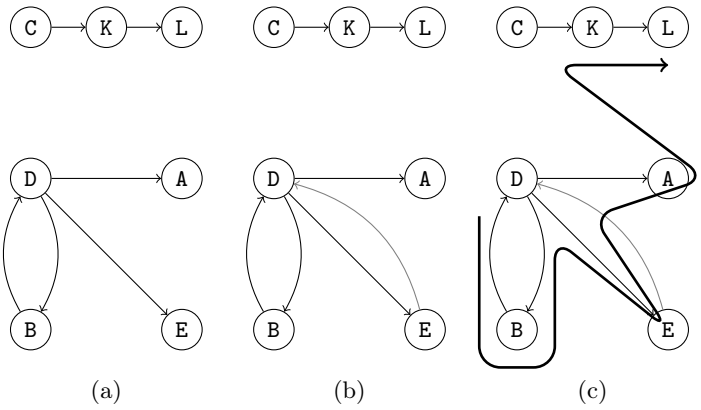


Fig. 2: 2-SCS can be solved in polynomial time. (a) de Bruijn graph of a set of strings {KL, DB, DE, CK, BD, DA}. (b) After adding an edge ED each weakly connected component contains an Eulerian path. (c) The string DBDEDACKL spelled by a path going through all the edges is a shortest superstring.

input strings have overlaps of size 2, the latter one corresponds to the case when the input strings do not overlap at all in a shortest superstring). Note that in the two extreme cases when $\mathrm{OPT}(\mathcal{S}) = n+2$ or $\mathrm{OPT}(\mathcal{S}) = 3n$ a shortest superstring can be easily found. Indeed, if $\mathrm{OPT}(\mathcal{S}) = n+2$ then $\mathcal{S}$ is the set of *all* substrings of length 3 of an unknown superstring of length $n+2$. Such a superstring can be found by traversing an Eulerian path in the de Bruijn graph of $\mathcal{S}$, $\mathrm{DG}(\mathcal{S})$. On the other hand, if $\mathrm{OPT}(\mathcal{S}) = 3n$ then the input strings just do not overlap with each other and any concatenation of them is a shortest superstring. The case $\mathrm{OPT}(\mathcal{S}) = n+2$ corresponds to the maximal possible compression while the case $\mathrm{OPT}(\mathcal{S}) = 3n$ corresponds to the minimal (i.e., zero) compression.

The algorithm proceeds as follows. We first construct the overlap graph $\mathrm{OG}(\mathcal{S})$ and find a 2/3-approximation of the maximum TSP path in it. Such a path provides a good approximation to $\mathrm{OPT}(\mathcal{S})$ in case $\mathrm{OPT}(\mathcal{S})$ is large.

We then construct the de Bruijn graph $\mathrm{DG}(\mathcal{S})$. Note that this graph can be viewed as the de Bruijn graph of a set of strings of length 2 over the alphabet Σ^2. Namely, in the original de Bruijn graph a string ABC is represented as an edge from AB to BC. This edge can be viewed as corresponding to the string (AB)(BC) of length 2 over the new alphabet. We then find a shortest superstring to this new set of 2-strings (recall that 2-SCS can be solved exactly in polynomial time) and translate the found solution back to the original problem. This gives a good approximation in case $\mathrm{OPT}(\mathcal{S})$ is small. The crucial fact is that if two input strings overlap a lot, then the corresponding 2-strings also overlap a lot and hence many overlaps are found by an algorithm for 2-SCS.

3.2 Formally

We are now ready to give all the details, see Algorithm 3.1. Note that the algorithm is quite easy to implement. Its only black-box part is a 2/3-approximation of MAX-ATSP. A recent algorithm achieving this ratio is due to Paluch et al. [18] and it is essentially based on finding a maximum weight matching. Thus, the running time of the presented algorithm is $O(n^3 \cdot \sum_{i=1}^{n} |s_i|) = O(n^4)$.

Algorithm 3.1 $(r^2 + r - 4)/(4r - 6)$-approximation algorithm r-SCS.

Input: $\mathcal{S} = \{s_1, \ldots, s_n\} \subseteq \Sigma^r$.
Output: A superstring of S that is at most $(r^2 + r - 4)/(4r - 6)$ times longer than a shortest superstring.

// first, find a long traveling salesman path in the overlap graph
1: let π be a 2/3-approximate maximum traveling salesman path in $\mathrm{OG}(\mathcal{S})$

// then, find a short rural postman path in the de Bruijn graph
2: let $\mathcal{S}' = \{s'_1, \ldots, s'_n\} \subseteq \Sigma_1^2$ be a set of 2-strings over the alphabet $\Sigma_1 = \Sigma^{r-1}$; s'_i is the 2-string consisting of prefix of s_i of length $r - 1$ and suffix of s_i of length $r - 1$
3: let π_1 be a shortest superstring for the set of 2-strings $\mathcal{S}'$
4: **return** the better one among π and π_1

Theorem 1. *Algorithm 3.1 finds an $\alpha(r)$-approximation for r-SCS where*

$$\alpha(r) = \frac{r^2 + r - 4}{4r - 6}\,.$$

Proof. Let H be a shortest Hamiltonian path in $\mathrm{OG}(\mathcal{S})$. Then clearly

$$\mathrm{OPT}(\mathcal{S}) = rn - w(H)\,,$$

where $w(H)$ is the weight of H. A 2/3-approximate maximum traveling salesman path has weight at least $2w(H)/3$. Thus, the permutation π gives a superstring of length at most $rn - 2w(H)/3$ (formally, to get a superstring from a permutation one just overlaps all the strings in this given order). The corresponding approximation ratio is

$$\frac{rn - 2w(H)/3}{rn - w(H)}\,. \tag{1}$$

Now let u denote the number of edges of weight at most $(r - 2)$ in H. Then the number of edges of weight exactly $(r - 1)$ in H is $(n - 1 - u)$. Then $w(H) \leq (r - 1)(n - 1 - u) + (r - 2)u$ and hence

$$u \leq (r - 1)(n - 1) - w(H)\,. \tag{2}$$

Note that

$$\mathrm{overlap}(s'_i, s'_j) = \begin{cases} 1 & \text{if } \mathrm{overlap}(s_i, s_j) = r - 1, \\ 0 & \text{otherwise.} \end{cases}$$

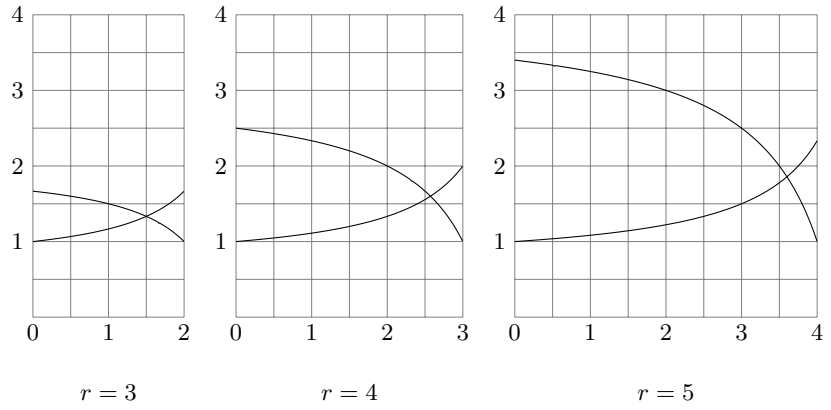


Fig. 3: Plots of $\frac{r-2x/3}{r-x}$ and $\frac{(r^2-2r+2)-(r-1)x}{r-x}$ for $r = 3, 4, 5$ and $0 \leq x \leq r-1$.

Since $\mathcal{S}'$ is a 2-SCS instance, a shortest superstring for $\mathcal{S}'$ has the maximal possible number of overlaps of size 1. This number is in turn equal to the maximal possible number of overlaps of size $r-1$ for $\mathcal{S}$. Since the number of overlaps of size $r-1$ in H is $(n-1-u)$ the length of a shortest superstring for $\mathcal{S}'$ has length at most $2n-(n-1-u) = n+u+1$. Hence π_1 gives a superstring of length at most $rn-(r-1)(n-1-u)$ in $\mathcal{S}$. Because of (2), this is at most

$$rn-(r-1)(n-1-(r-1)(n-1)+w(H)) < (r^2-2r+2)n-(r-1)w(H)\,.$$

The corresponding approximation ratio is

$$\frac{(r^2-2r+2)n-(r-1)w(H)}{rn-w(H)}\,. \tag{3}$$

From (1) and (3) and a simple observation that $0 \leq w(H)/n \leq (r-1)$ we conclude that the approximation ratio of the constructed algorithm is

$$\alpha(r) = \max_{0\leq x\leq r-1}\left\{\min\left\{\frac{r-2x/3}{r-x}, \frac{(r^2-2r+2)-(r-1)x}{r-x}\right\}\right\}\,.$$

Fig. 3 shows plots of the considered functions for $r = 3, 4, 5$.

By taking the derivatives it is easy to see that the former function increases while the latter one decreases on $[0, r-1]$. This means that the maximum of their minimum is attained at x where they meet, namely

$$x = \frac{r^2-3r+2}{r-5/3}\,.$$

Plugging in this x gives

$$\alpha(r) = \frac{r^2+r-4}{4r-6}\,.$$

□

Acknowledgements

Research is partially supported by Russian Foundation for Basic Research (11-01-00760-a, 12-01-31057), RAS Program for Fundamental Research, Grant of the President of Russian Federation (NSh-3229.2012.1), the Ministry of Education and Science of the Russian Federation (8216), the Government of the Russian Federation (11.G34.31.0018), and Computer Science Club scholarship.

Also, we would like to thank the anonymous referees for many valuable comments.

References

1. Armen, C., Stein, C.: A $2\frac{3}{4}$-Approximation Algorithm for the Shortest Superstring Problem. Tech. rep., Dartmouth College, Hanover, NH, USA (1994)
2. Armen, C., Stein, C.: Improved length bounds for the shortest superstring problem. In: Algorithms and Data Structures, LNCS, vol. 955, pp. 494–505. Springer Berlin / Heidelberg (1995)
3. Armen, C., Stein, C.: A $2\frac{2}{3}$-Approximation for the Shortest Superstring Problem. In: Hirschberg, D., Myers, G. (eds.) Combinatorial Pattern Matching, LNCS, vol. 1075, pp. 87–101. Springer Berlin / Heidelberg (1996)
4. Blum, A., Jiang, T., Li, M., Tromp, J., Yannakakis, M.: Linear approximation of shortest superstrings. In: Proceedings of the twenty-third annual ACM symposium on Theory of computing. pp. 328–336. STOC'91, ACM, New York, NY, USA (1991)
5. Breslauer, D., Jiang, T., Jiang, Z.: Rotations of periodic strings and short superstrings. J. Algorithms 24(2), 340–353 (Aug 1997)
6. Christofides, N., Campos, V., Corberan, A., Mota, E.: An algorithm for the Rural Postman problem on a directed graph. In: Netflow at Pisa, Mathematical Programming Studies, vol. 26, pp. 155–166. Springer Berlin Heidelberg (1986)
7. Crochemore, M., Cygan, M., Iliopoulos, C., Kubica, M., Radoszewski, J., Rytter, W., Walen, T.: Algorithms for three versions of the shortest common superstring problem. In: Proceedings of the 21st annual conference on Combinatorial pattern matching. pp. 299–309. CPM'10, Springer-Verlag (2010)
8. Czumaj, A., Gasieniec, L., Piotrow, M., Rytter, W.: Parallel and sequential approximation of shortest superstrings. In: Algorithm Theory - SWAT'94, LNCS, vol. 824, pp. 95–106. Springer Berlin / Heidelberg (1994)
9. Eiselt, H.A., Gendreau, M., Laporte, G.: Arc Routing Problems, Part II: The Rural Postman Problem. Operations Research 43(3), 399–414 (1995)
10. Gallant, J., Maier, D., Storer, J.A.: On finding minimal length superstrings. Journal of Computer and System Sciences 20(1), 50–58 (1980)
11. Groves, G., van Vuuren, J.: Efficient heuristics for the Rural Postman Problem. ORiON 21(1), 33–51 (Jun 2005)
12. Kaplan, H., Lewenstein, M., Shafrir, N., Sviridenko, M.: Approximation Algorithms for Asymmetric TSP by Decomposing Directed Regular Multigraphs. J. ACM 52, 602–626 (July 2005)
13. Kaplan, H., Shafrir, N.: The greedy algorithm for shortest superstrings. Inf. Process. Lett. 93(1), 13–17 (Jan 2005)
14. Karpinski, M., Schmied, R.: Improved Lower Bounds for the Shortest Superstring and Related Problems. CoRR abs/1111.5442v3 (2012)

15. Kosaraju, S.R., Park, J.K., Stein, C.: Long tours and short superstrings. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science. pp. 166–177. SFCS'94, IEEE Computer Society, Washington, DC, USA (1994)
16. Mucha, M.: Lyndon Words and Short Superstrings. In: Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA'13, Society for Industrial and Applied Mathematics (2013)
17. Ott, S.: Lower Bounds for Approximating Shortest Superstrings over an Alphabet of Size 2. In: Graph-Theoretic Concepts in Computer Science, LNCS, vol. 1665, pp. 55–64. Springer Berlin / Heidelberg (1999)
18. Paluch, K., Elbassioni, K., van Zuylen, A.: Simpler Approximation of the Maximum Asymmetric Traveling Salesman Problem. In: STACS '12. LIPIcs, vol. 14, pp. 501–506 (2012)
19. Pevzner, P.A., Tang, H., Waterman, M.S.: An Eulerian path approach to DNA fragment assembly. Proc. Natl. Acad. Sci. 98(17), 9748–9753 (Aug 2001)
20. Sahni, S., Gonzalez, T.: P-Complete Approximation Problems. J. ACM 23, 555–565 (July 1976)
21. Sweedyk, Z.: $2\frac{1}{2}$-Approximation Algorithm for Shortest Superstring. SIAM J. Comput. 29(3), 954–986 (Dec 1999)
22. Tarhio, J., Ukkonen, E.: A greedy approximation algorithm for constructing shortest common superstrings. Theoretical Computer Science 57(1), 131–145 (1988)
23. Teng, S.H., Yao, F.: Approximating shortest superstrings. In: Proceedings of the 1993 IEEE 34th Annual Foundations of Computer Science. pp. 158–165. SFCS'93, IEEE Computer Society, Washington, DC, USA (1993)
24. Turner, J.S.: Approximation algorithms for the shortest common superstring problem. Information and Computation 83(1), 1–20 (1989)
25. Vassilevska, V.: Explicit Inapproximability Bounds for the Shortest Superstring Problem. In: Mathematical Foundations of Computer Science 2005, LNCS, vol. 3618, pp. 793–800. Springer Berlin / Heidelberg (2005)