Solving SCS for bounded length strings in fewer than 2^n steps

Alexander Golovnev * Alexander S. Kulikov † Ivan Mihajlin ‡

Abstract

It is still not known whether a shortest common superstring (SCS) of n input strings can be found faster than in $O^*(2^n)$ time ($O^*(\cdot)$ suppresses polynomial factors of the input length). In this short note, we show that for any constant r, SCS for strings of length at most r can be solved in time $O^*(2^{(1-c(r))n})$ where $c(r) = (1+2r^2)^{-1}$. For this, we introduce so-called hierarchical graphs that allow us to reduce SCS on strings of length at most r to the directed rural postman problem on a graph with at most $k = (1-c(r))n$ weakly connected components. One can then use a recent $O^*(2^k)$ time algorithm by Gutin, Wahlström, and Yeo.

1 Introduction

The *shortest common superstring problem* (*SCS*) asks for a shortest string which contains each of the given strings $s_1, \ldots, s_n$ as a substring. The problem has many practical applications including genome assembly and sparse matrix compression. By the *r-superstring problem* (*r-SCS*) we denote the SCS problem for the special case where all input strings have length at most r. Both SCS over the binary alphabet and r-SCS (for $r \geq 3$) are known to be NP-hard optimization problems [11]. Although approximation algorithms for SCS are widely studied (currently, the best known approximation ratio is $2\frac{11}{23}$ due to Mucha [21]), it is still not known whether SCS can be solved exactly in fewer than $O^*(2^n)$ steps[1]. At the same time an easy reduction of SCS to the traveling salesman problem (TSP) gives an algorithm solving SCS in $O^*(2^n)$ time and polynomial space [17, 16, 1]. In this note, we show that for any constant r, r-SCS can be solved in time $O^*\left(2^{(1-\frac{1}{2r^2+1})n}\right)$. The result is achieved by combining a new combinatorial structure called hierarchical graphs (inspired by de Bruijn graphs) with a recent algorithm solving the directed rural postman problem in $O^*(2^k)$ time where k is the number of weakly connected components by Gutin, Wahlström, and Yeo [13].

Thus, the main result shows that SCS can be solved faster than $O^*(2^n)$ when input strings are short. At the same time the other natural special case of SCS when the alphabet size is small is as hard as the general case: Vassilevska [23] proved that an $O^*(c^n)$-algorithm for SCS over the binary alphabet implies an $O^*(c^n)$-algorithm for the general case.

Our initial motivation for studying this problem was the existence of similar algorithms for other very well-known NP-hard problems — say, the satisfiability problem (SAT), the maximum satisfiability problem (MAX-SAT), the coloring problem, and the traveling salesman problem. Despite many efforts the best known algorithms for the general versions of these problems have running time $O^*(2^n)$ (n being the number of variables/vertices). For SAT and MAX-SAT, $O^*(2^n)$ is the running time of an exhaustive search. For TSP, $O^*(2^n)$ bound is proved using the dynamic programming method by Bellman [3] and Held and Karp [14]. For coloring, $O^*(2^n)$ bound was proved only recently using the inclusion-exclusion principle by Björklund, Husfeldt and Koivisto [6]. At the same time better upper bounds are known for various special cases of these problems:

*New York University

†St. Petersburg Department of Steklov Institute of Mathematics. Research is partially supported by the Program for development of Centers of Advanced Studies of Ministry of Tele- and Mass Communications of the Russian Federation.

‡St. Petersburg Academic University. Supported by the Government of the Russian Federation (Resolution No. 220).

[1] $O^*(c^n)$ suppresses polynomial factors of the input length. We omit the star if we round up c.

- SAT: $O(1.308^n)$ for 3-SAT [15]; $O^*((2-2/k)^n)$ for k-SAT [22, 20];
- MAX-SAT: $O(1.731^n)$ for MAX-2-SAT [24, 18], $O(1.109^n)$ for $(n,3)$-MAX-2-SAT [19]; $O((2-\varepsilon)^n)$ for formulas with constant clause density [10, 19];
- TSP: $O(1.2186^n)$ for cubic graphs [7]; $O((2-\varepsilon)^n)$ for graphs of bounded maximum/average degree [4, 9];
- Coloring: $O(1.3289^n)$ for 3-coloring [2]; $O((2-\varepsilon)^n)$ for graphs of bounded maximum degree [5];
- SCS: $O(1.443^n)$ for strings of length 3 [12].

Improving the $O^*(2^n)$ bound for all these problems (as well as for SCS) remains a challengeable open problem.

2 General Setting

2.1 Strings and Superstrings

By $u \sqsupset v$ ($u \sqsubset v$) we denote that u is a suffix (prefix) of v. For strings s and t, by $\text{overlap}(s,t)$ we denote the longest suffix of s that is also a prefix of t. By $\text{prefix}(s,t)$ we denote the first $|s| - |\text{overlap}(s,t)|$ symbols of s. Similarly, $\text{suffix}(s,t)$ is the last $|t| - |\text{overlap}(s,t)|$ symbols of t. By $\text{pref}(s)$ and $\text{suf}(s)$ we denote, respectively, the first and the last $|s|-1$ symbols of s. The empty string is denoted by ε.

Throughout the paper by $\mathcal{S} = \{s_1, \ldots, s_n\}$ we denote a set of n input strings, each of length at most $r = O(1)$. We assume that no input string is a substring of another (such a substring can be removed from $\mathcal{S}$ while solving SCS for $\mathcal{S}$ on the preprocessing stage). Note that SCS is a permutation problem: to find a shortest string containing all s_i's in a given order one just overlaps the strings in this order. This simple observation makes many connections to other permutation problems, including different versions of TSP.

2.2 Graphs and Walks

By a *path* in a directed graph we mean a path with no repeated vertices. We use the term *walk* for a path in which vertices may be repeated. A *closed walk* is a walk whose first vertex is the same as the last.

In the main result of this note we reduce SCS to the *directed rural postman problem* (*DRPP*). In this problem one is given a weighted directed multigraph $G = (V, A)$ together with a set of arcs $R \subseteq A$ and the goal is to find a shortest closed walk going through all the arcs from R. The arcs R are called *required* while all the remaining arcs are called *optional*. Although DRPP is NP-hard in general case, it can be solved in polynomial time if the arcs from R form a single weakly connected component [8] (weakly connected components of a directed graph are connected components in this graph with all directed arcs replaced by undirected edges). We use also the following recent result by Gutin, Wahlström and Yeo [13].

Theorem 1. *Let $G = (V, A)$ be a weighted directed multigraph, $R \subseteq A$ be a subset of arcs, $l = \text{poly}(|V|)$, then there exists a randomized algorithm with false negatives checking whether the length of a shortest closed walk going through all the arcs from R is at most l in time $O^*(2^k)$, where k is the number of weakly connected components in the subgraph of G induced by R.*

3 Hierarchical Graphs

Definition 1 (hierarchical graph)**.** *A* hierarchical graph *$HG_{\mathcal{S}} = (V, A)$ of $\mathcal{S}$ is a weighted directed graph defined as follows:*

- *The set of vertices V consists of all prefixes and suffixes (including the empty string ε) of the strings from $\mathcal{S}$.*
- *For two such strings $u, v \in V$, $(u, v) \in A$ when either*

– *u is a prefix of v of length $|v|-1$ (i.e., $u = \mathrm{pref}(v)$); in this case the weight $w(u,v) = 1$ and (u,v) is called an* up-arc*, or*

– *v is a suffix of u of length $|u|-1$ (i.e., $v = \mathrm{suf}(u)$); in this case the weight $w(u,v) = 0$ and (u,v) is called a* down-arc*.*

Figure 1 gives an example of the hierarchical graph as well as shows that the terminology of up- and down-arcs comes from placing all the strings of the same length at the same layer where the i-th layer contains strings of length i. For an i-th layer the $(i-1)$-th layer is called *previous* while the $(i+1)$-th layer is called *next*. By an *up-path* (resp., *down-path*) we denote a path containing up-arcs (down-arcs) only. A path $\texttt{d} \to \texttt{db} \to \texttt{dbb}$ on Fig. 1 is an example of an up-path, a path $\texttt{abb} \to \texttt{bb} \to \texttt{b}$ is a down-path.

What we are looking for in this graph is a shortest walk from ε to ε going through all the vertices from $\mathcal{S}$. It is not difficult to see that the length of a walk from ε to ε equals the length of the string spelled by this walk. This is just because each arc going up has weight 1 and adds one symbol to the current string. For example, in the graph of Fig. 1 a walk $\varepsilon \to \texttt{b} \to \texttt{ba} \to \texttt{bab} \to \texttt{ab} \to \texttt{abc} \to \texttt{abca} \to \texttt{bca} \to \texttt{ca} \to \texttt{a} \to \varepsilon$ has length 5 and spells a string $\texttt{babca}$ of length 5 in a natural way.

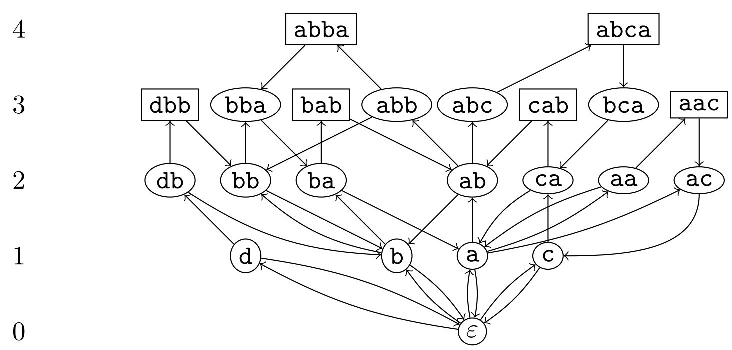

Figure 1: The hierarchical graph for $\mathcal{S} = \{\texttt{abba}, \texttt{abca}, \texttt{dbb}, \texttt{bab}, \texttt{cab}, \texttt{aac}\}$. The strings from $\mathcal{S}$ are given in rectangles.

Note that each string from $\mathcal{S}$ has exactly one incoming and one outgoing arc in $HG_{\mathcal{S}}$. Denote the set of all these arcs by R. Any optimal walk must go through all the arcs of R so we call these arcs *required* (see Fig. 2(a)). Formally, $R = \{(\mathrm{pref}(s), s) \colon s \in \mathcal{S}\} \cup \{(s, \mathrm{suf}(s)) \colon s \in \mathcal{S}\}$. The resulting problem is an instance of the directed rural postman problem with the only exception: optimal walk must go through ε. To guarantee that a rural walk starts in ε and ends in ε we add a self-loop on ε of weight 0 to the set of required arcs.

For any two vertices u and v of this graph such that none of them is a substring of the other one there is a *natural* path from u to v: it first goes down from u to $\mathrm{overlap}(u,v)$ and then goes up to v (see Fig. 3). We call a rural walk *normal* if between visiting two consecutive vertices from $\mathcal{S}$ it alternates directions only once (i.e., any subpath between two consecutive vertices from $\mathcal{S}$ is a natural path). It is easy to see that there always exists a normal optimal walk (recall that any $s \in \mathcal{S}$ is not a substring of any other $s' \in \mathcal{S}$).

If all overlaps of a particular string $s \in \mathcal{S}$ (with other strings from $\mathcal{S}$) are short we know for sure that any optimal rural walk has a long down-path out of s. Returning to the example of Fig. 1, it is easy to see that no string from $\mathcal{S}$ starts from $\texttt{bb}$. This means that any optimal rural walk in $HG_{\mathcal{S}}$ must go down from $\texttt{dbb}$ to at least $\texttt{b}$. We formalize this intuition in the definition below.

Definition 2 (extended set of required arcs)**.** *For a string $s \in \mathcal{S}$, denote by $\mathrm{maxpref}_{\mathcal{S}}(s)$ (resp., $\mathrm{maxsuf}_{\mathcal{S}}(s)$) the longest prefix (resp., suffix) of s which is also a suffix (prefix) of some other string $s' \in \mathcal{S}$. Clearly,*

$$|\mathrm{maxpref}_{\mathcal{S}}(s)| = \max\{|\mathrm{overlap}(s', s)| \colon s' \in \mathcal{S} \setminus \{s\}\},$$
$$|\mathrm{maxsuf}_{\mathcal{S}}(s)| = \max\{|\mathrm{overlap}(s, s')| \colon s' \in \mathcal{S} \setminus \{s\}\}.$$

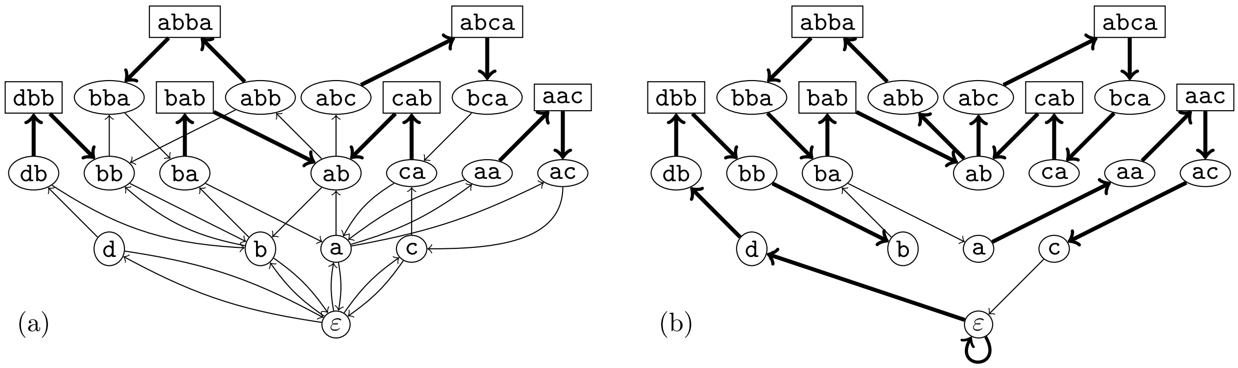

Figure 2: (a) The set R of required arcs is shown in bold. (b) An optimal superstring dbbabcabbaac defines a walk of length 12 in HG. Note that both the superstring and the optimal walk are defined by a permutation $\sigma = (\texttt{dbb}, \texttt{bab}, \texttt{abca}, \texttt{cab}, \texttt{abba}, \texttt{aac})$. The extended set of required arcs ER is shown in bold.

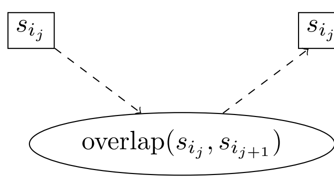

Figure 3: A natural path between vertices s_{i_j} and $s_{i_{j+1}}$.

Then an extended set of required arcs $ER \subseteq E(HG_{\mathcal{S}})$ *is the set of required arcs R plus the following set of arcs:*

- *the up-path from* $\mathrm{maxpref}_{\mathcal{S}}(s)$ *to* s*: all arcs* (u, v) *where* $u, v \sqsubset s$, $|u| = |v| - 1$ *and* $|u| \geq |\mathrm{maxpref}_{\mathcal{S}}(s)|$*;*
- *the down-path from* s *to* $\mathrm{maxsuf}_{\mathcal{S}}(s)$*: all arcs* (u, v) *where* $u, v \sqsupset s$, $|u| = |v| + 1$ *and* $|v| \geq |\mathrm{maxsuf}_{\mathcal{S}}(s)|$*;*
- *the loop* $(\varepsilon, \varepsilon)$ *of weight* 0*.*

E.g., for $\mathcal{S} = \{\texttt{abba}, \texttt{abca}, \texttt{dbb}, \texttt{bab}, \texttt{cab}, \texttt{aac}\}$, $\mathrm{maxpref}_{\mathcal{S}}(\texttt{abca}) = \texttt{ab}$, $\mathrm{maxsuf}_{\mathcal{S}}(\texttt{abca}) = \texttt{ca}$, $\mathrm{maxpref}_{\mathcal{S}}(\texttt{dbb}) = \varepsilon$, $\mathrm{maxsuf}_{\mathcal{S}}(\texttt{dbb}) = \texttt{b}$.

Lemma 1. *The length of an optimal superstring of a set of strings* $\mathcal{S}$ *is equal to the length of an optimal rural postman closed walk in* $HG_{\mathcal{S}}$ *where the required arcs are* ER*.*

Proof. Consider an optimal rural closed walk w and represent it as a sequence of vertices $v_0 = \varepsilon, v_1, \ldots, v_k = \varepsilon$. This walk spells a string s in a natural way: initially, set $s = \varepsilon$ and start traversing the walk; each time when it goes up (i.e., $|v_i| = |v_{i-1}| + 1$) add the corresponding symbol to s. This way we preserve the following two invariants:

- the length of the current string equals the length of the traversed subwalk;
- when at a vertex v_i, the current string s contains v_i as a suffix.

Since w goes through all the strings from $\mathcal{S}$ the resulting string s is a superstring of $\mathcal{S}$. Clearly, the length of s equals the length of w. Thus, the length of an optimal superstring does not exceed the length of an optimal rural walk.

For the reverse direction, consider a superstring s for $\mathcal{S}$. It defines an order $s_{i_1}, \ldots, s_{i_n}$ of the strings from $\mathcal{S}$. Consider now the following normal rural walk: it starts at ε, goes up to s_{i_1}, then for all $j = 1, \ldots, n-1$ it goes down from s_{i_j} to $\mathrm{overlap}(s_{i_j}, s_{i_{j+1}})$ and then goes up to $s_{i_{j+1}}$, then it goes down from s_{i_n} to ε, and finally it goes through the loop $(\varepsilon, \varepsilon)$. It is easy to see that the length of the resulting closed walk equals the length of s. It is also a valid rural postman walk since for each string s_{i_j}

$$\begin{aligned} |\,\mathrm{overlap}(s_{i_{j-1}}, s_{i_j})| &\leq |\,\mathrm{maxpref}_{\mathcal{S}}(s_{i_j})|\,, \\ |\,\mathrm{overlap}(s_{i_j}, s_{i_{j+1}})| &\leq |\,\mathrm{maxsuf}_{\mathcal{S}}(s_{i_j})|\,. \end{aligned}$$

Hence the walk necessarily traverses all the arcs from ER. Thus, the length of an optimal rural closed walk is not greater then the length of an optimal superstring. □

Definition 3 (bottom vertex)**.** *A vertex v in $HG_{\mathcal{S}}$ is called a* bottom vertex *if*

$$\{(u,v) \in ER : |u| = |v| - 1\} = \{(v,u) \in ER : |u| = |v| - 1\} = \emptyset \text{ and}$$

$$|\{(u,v) \in ER : |u| = |v| + 1\}| + |\{(v,u) \in ER : |u| = |v| + 1\}| \geq 1\,.$$

In other words, v is not connected to the previous (i.e., $(|v|-1)$-th) layer by the arcs of ER, but is connected to the next (i.e., $(|v|+1)$-th) layer.

Lemma 2. $V_b = \{\mathrm{maxpref}_{\mathcal{S}}(s), \mathrm{maxsuf}_{\mathcal{S}}(s) \colon s \in \mathcal{S}\}$.

Proof. Clearly, any bottom vertex is either $\mathrm{maxpref}_{\mathcal{S}}(s)$ or $\mathrm{maxsuf}_{\mathcal{S}}(s)$ for some $s \in \mathcal{S}$. For the other direction, consider a vertex $v = \mathrm{maxsuf}_{\mathcal{S}}(s)$ and assume that it has an incoming up-arc $(u,v) \in ER$. This arc must lie on an up-path from $\mathrm{maxpref}_{\mathcal{S}}(t)$ to t for some $t \in \mathcal{S}$. But then $v \sqsubset t$ and $v \sqsupset s$ and v is strictly longer than $\mathrm{maxpref}_{\mathcal{S}}(t)$ which contradicts to the definition of $\mathrm{maxpref}_{\mathcal{S}}(t)$. By a similar argument one can show that v does not have outgoing down-arcs. This shows that $\mathrm{maxsuf}_{\mathcal{S}}(s)$ is indeed a bottom vertex. By the same reason, $\mathrm{maxpref}_{\mathcal{S}}(s)$ is also a bottom vertex. □

E.g., for the set $\mathcal{S}$ from Fig. 1, $\{\mathrm{maxpref}_{\mathcal{S}}(s) \colon s \in \mathcal{S}\} = \{\varepsilon, \mathtt{ba}, \mathtt{ab}, \mathtt{ca}, \mathtt{a}\}$ and $\{\mathrm{maxsuf}_{\mathcal{S}}(s) \colon s \in \mathcal{S}\} = \{\mathtt{b}, \mathtt{ab}, \mathtt{ca}, \mathtt{ba}, \mathtt{c}\}$.

Definition 4 (good vertex)**.** *A bottom vertex is called* good *if it is not a substring of any other bottom vertex. The set of all good vertices is denoted by V_g.*

In Fig. 2(b) the bottom vertices are ε, b, ba, ab, ca, a, c. Among them, ba, ab, and ca are good.

Note that a good vertex t is a meeting point of a down-path from $s \in \mathcal{S}$ to $t = \mathrm{maxsuf}_{\mathcal{S}}(s)$ and an up-path from $t = \mathrm{maxpref}_{\mathcal{S}}(s')$ to $s' \in \mathcal{S}$. Indeed, since a good vertex is a bottom vertex, it has either a down-path $s \rightsquigarrow t$ or an up-path $t \rightsquigarrow s'$ in ER. Consider the case that the path $s \rightsquigarrow t$ is in ER (the other case is symmetric). If the entire path $t \rightsquigarrow s' = \{t = v_0 \to v_1 \to \ldots \to v_k = s'\}$ is in ER then we are done. Assume, to the contrary, that there is a vertex v_i in the path such that $(v_{i-1}, v_i) \notin ER$ while the path $v_i \rightsquigarrow s' \in ER$. In order to get a contradiction, we want to show that v_i is a bottom vertex (this cannot be the case since t is a substring of v_i while by definition t is not a substring of any bottom vertex). Indeed, there is an arc from v_i to a vertex v_{i+1} from the next layer, but there are no connections to the previous layer:

- the up-arc $(v_{i-1} = \mathrm{pref}(v_i), v_i) \notin ER$ by the assumption;
- the down-arc $(v_i, \mathrm{suf}(v_i)) \notin ER$ because any down-path from an input string to v_i stops at v_i or earlier (more formally, if $(v_i, \mathrm{suf}(v_i)) \in ER$ then there exists an input string $s_0 \in \mathcal{S}$ such that $|\,\mathrm{maxsuf}_{\mathcal{S}}(s_0)| \leq |\,\mathrm{suf}(v_i)| = |v_i| - 1$; at the same time $|\,\mathrm{maxsuf}_{\mathcal{S}}(s_0)| \geq |\,\mathrm{overlap}(s_0, s')| \geq |v_i|$, a contradiction).

Lemma 3. $r^2|V_g| \geq |V_b|$.

Proof. Recall that a bottom vertex v is not good iff there is another bottom vertex u such that v is a substring of u. This allows to define recursively the following mapping $f\colon V_b \to V_g$: if $v \in V_b$ is good then $f(v) = v$; otherwise take a vertex $u \in V_b$ such that v is a substring of u and set $f(v) = f(u)$ (this is feasible since $|u| > |v|$). I.e., we go up from v till we reach a good vertex. Now note that for any $v \in V_b$, v is a substring of $f(v)$. Since each vertex has at most r^2 substrings we have $r^2|V_g| \geq |V_b|$. □

Theorem 2. *The set ER of extended required arcs consists of at most $(1 - \frac{1}{2r^2+1})n$ weakly connected components.*

Proof. Let k be the total number of weakly connected components and m_i be the number of weakly connected components that contain exactly i strings from $\mathcal{S}$. Then

$$\sum_{i=1}^{n} m_i = k \text{ and } \sum_{i=1}^{n} i m_i = n\,.$$

Since only bottom vertices have no connections to the previous layer, each weakly connected component contains at least one bottom vertex, hence $k \leq |V_b|$.

Also, each weakly connected component containing i input strings contains at most i good vertices. Indeed, a good vertex t is a meeting point of a down-path from $s \in \mathcal{S}$ and an up-path to $s' \in \mathcal{S}$. At the same time any $s \in \mathcal{S}$ produces no more than two good vertices (one corresponding to $\mathrm{maxpref}_{\mathcal{S}}(s)$ and one corresponding to $\mathrm{maxsuf}_{\mathcal{S}}(s)$; recall Lemma 2). Hence $\sum_{i=2}^{n} i m_i \geq |V_g|$ (clearly a component with only one input string does not contain good vertices at all).

Using these estimates and applying Lemma 3 we get

$$n = \sum_{i=1}^{n} i m_i = k + \sum_{i=1}^{n} (i-1) m_i \geq k + \sum_{i=2}^{n} \frac{i m_i}{2} \geq k + \frac{|V_g|}{2} \geq k + \frac{|V_b|}{2r^2} \geq k + \frac{k}{2r^2} = k\left(1 + \frac{1}{2r^2}\right)\,.$$

□

4 Main Result

Theorem 3. *There exists a randomized algorithm solving r-SCS on n strings in time $O^*\left(2^{\left(1-\frac{1}{2r^2+1}\right)n}\right)$.*

Proof. Lemma 1 tells that to find a shortest superstring for $\mathcal{S}$ it is enough to find a shortest rural postman closed walk in $HG_{\mathcal{S}}$ for a set of required arcs ER. Theorem 2 guarantees that the number k of weakly connected components of ER is at most $(1 - \frac{1}{1+2r^2})n$. Finally, Theorem 1 shows that one can check whether such a graph contains a closed walk of total length $l = \mathrm{poly}(|V|)$ going through all required arcs in time $O^*(2^k)$. In our case l is indeed $\mathrm{poly}(|V|)$ since the optimal length of a superstring of $\mathcal{S}$ does not exceed rn. □

5 Further Directions

The natural next step is to solve the general version SCS in $O^*((2-\epsilon)^n)$. It would also be interesting to show hardness of SCS (under Strong Exponential Time Hypothesis or, e.g., by reducing TSP with n vertices to SCS with n strings).

References

[1] Eric Bax and Joel Franklin. A finite-difference sieve to count paths and cycles by length. *Inf. Process. Lett.*, 60:171–176, November 1996.

[2] Richard Beigel and David Eppstein. 3-coloring in time $O(1.3289^n)$. *Journal of Algorithms*, 54(2):168–204, 2005.

[3] Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9:61–63, January 1962.

[4] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. The travelling salesman problem in bounded degree graphs. In *Automata, Languages and Programming*, volume 5125 of *LNCS*, pages 198–209. Springer Berlin / Heidelberg, 2008.

[5] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Trimmed moebius inversion and graphs of bounded degree. *Theory of Computing Systems*, 47(3):637–654, 2010.

[6] Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009.

[7] Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. In *Automata, Languages, and Programming*, pages 196–207. Springer, 2013.

[8] Nicos Christofides, V. Campos, A. Corberan, and E. Mota. An algorithm for the Rural Postman problem on a directed graph. In *Netflow at Pisa*, volume 26 of *Mathematical Programming Studies*, pages 155–166. Springer Berlin Heidelberg, 1986.

[9] Marek Cygan and Marcin Pilipczuk. Faster exponential-time algorithms in graphs of bounded average degree. In *Automata, Languages, and Programming*, volume 7965 of *Lecture Notes in Computer Science*, pages 364–375. Springer Berlin Heidelberg, 2013.

[10] Evgeny Dantsin and Alexander Wolpert. MAX-SAT for formulas with constant clause density can be solved faster than in $O(2^n)$ time. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing*, volume 4121 of *LNCS*, pages 266–276, 2006.

[11] John Gallant, David Maier, and James A. Storer. On finding minimal length superstrings. *Journal of Computer and System Sciences*, 20(1):50–58, 1980.

[12] Alexander Golovnev, Alexander S. Kulikov, and Ivan Mihajlin. Solving 3-superstring in $3^{n/3}$ time. In *Mathematical Foundations of Computer Science 2013*, volume 8087 of *Lecture Notes in Computer Science*, pages 480–491. Springer Berlin Heidelberg, 2013.

[13] Gregory Gutin, Magnus Wahlström, and Anders Yeo. Parameterized rural postman and conjoining bipartite matching problems. *arXiv preprint arXiv:1308.2599*, 2013.

[14] Michael Held and Richard M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962.

[15] Timon Hertli. 3-SAT faster and simpler - unique-SAT bounds for PPSZ hold in general. In *Foundations of Computer Science (FOCS)*, pages 277 –284, oct. 2011.

[16] Richard M. Karp. Dynamic programming meets the principle of inclusion and exclusion. *Operations Research Letters*, 1(2):49–51, 1982.

[17] Samuel Kohn, Allan Gottlieb, and Meryle Kohn. A generating function approach to the traveling salesman problem. In *ACN'77: Proceedings of the 1977 annual conference*, pages 294–300, New York, NY, USA, 1977.

[18] Mikko Koivisto. Optimal 2-constraint satisfaction via sum–product algorithms. *Information processing letters*, 98(1):24–28, 2006.

[19] Alexander S. Kulikov and Konstantin Kutzkov. New upper bounds for the problem of maximal satisfiability. *Discrete Mathematics and Applications*, 19:155–172, 2009.

[20] Robin A. Moser and Dominik Scheder. A full derandomization of Schöning's k-SAT algorithm. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 245–252. ACM, 2011.

[21] Marcin Mucha. Lyndon words and short superstrings. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'13, pages 958–972. Society for Industrial and Applied Mathematics, 2013.

[22] Uwe Schöning. A probabilistic algorithm for k-SAT based on limited local search and restart. *Algorithmica*, 32(4):615–623, 2002.

[23] Virginia Vassilevska. Explicit inapproximability bounds for the shortest superstring problem. In *Mathematical Foundations of Computer Science 2005*, volume 3618 of *LNCS*, pages 793–800. Springer Berlin / Heidelberg, 2005.

[24] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.