Solving 3-Superstring in $3^{n/3}$ Time

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. In the shortest common superstring problem (SCS) one is given a set $s_1, \ldots, s_n$ of n strings and the goal is to find a shortest string containing each s_i as a substring. While many approximation algorithms for this problem have been developed, it is still not known whether it can be solved exactly in fewer than 2^n steps. In this paper we present an algorithm that solves the special case when all of the input strings have length 3 in time $3^{n/3}$ and polynomial space. The algorithm generates a combination of a de Bruijn graph and an overlap graph, such that a SCS is then a shortest directed rural postman path (DRPP) on this graph. We show that there exists at least one optimal DRPP satisfying some natural properties. The algorithm works basically by exhaustive search, but on the reduced search space of such paths of size $3^{n/3}$.

1 Introduction

The *shortest common superstring problem* (SCS) is: given a set $\{s_1, \ldots, s_n\}$ of n strings, find a shortest string containing each s_i as a substring (w.l.o.g., we assume that no input string is a subtstring of another). The problem is known to be NP-hard and has many practical applications including data storage, data compression, and genome assembly. For this reason, approximation algorithms for SCS are widely studied. For a long time the best known approximation ratio was 2.5 by Sweedyk [26] (the same bound also follows from 2/3-approximation for MAX-ATSP [15,24]). Very recently the bound was improved to $2\frac{11}{23}$ by Mucha [23]. The best known inapproximability ratio (under the P $\neq$ NP assumption) is $\frac{345}{344}$ by Karpinski and Schmied [17].

At the same time it is not known whether SCS can be solved in fewer than $O^*(2^n)$ steps ($O^*(\cdot)$ suppresses polynomial factors of input length). Note that SCS is a permutation problem: to find a string containing all s_i's in a given order one just overlaps the strings in this order. Thus, the trivial algorithm requires $O^*(n!)$ time. Now consider the following *suffix graph* of the given set of strings: the set of vertices is $\{s_1, \ldots, s_n\}$, vertices s_i and s_j are joined by an arc of weight $|\mathrm{suffix}(s_i, s_j)|$ where $\mathrm{suffix}(s_i, s_j)$ is the shortest string such that s_j is a suffix of $s_i \circ \mathrm{suffix}(s_i, s_j)$ (where $\circ$ denotes concatenation). SCS can be solved by finding a shortest traveling salesman path (TSP) in this graph. For TSP, the classical dynamic programming based $O^*(2^n)$ algorithm discovered by Bellman [2] and independently by Held and Karp [12] is well-known.

There are two natural special cases of SCS: the case when the size of the alphabet is bounded by a constant and the case when all input strings have length r. The latter case is called *r-SCS*. Note that when both these parameters are bounded the problem degenerates as the number of possible input strings is then also bounded by a constant. It is known that both SCS over the binary alphabet and 3-SCS are NP-hard, while 2-SCS can be solved in linear time [10] and 2-SCS with multiplicities (where each input string is given together with the number of its occurrences in a superstring) can be solved in quadratic time [7]. Vassilevska [27] showed that SCS over the binary alphabet cannot be much easier than the general case. Namely, she provided a polynomial-time reduction of general SCS to SCS over the binary alphabet that preserves the number of input strings. It implies that α-approximation of SCS over the binary alphabet is not easier than α-approximation of general SCS. It also means that an $O^*(c^n)$-algorithm for SCS over the binary alphabet implies an $O^*(c^n)$-algorithm for the general case. Hence SCS for smaller size alphabet cannot be much easier. Our result suggests that SCS for shorter strings can actually be easier to solve.

In this paper we present an algorithm solving a special case when all of the input strings have length 3 in time $O^*(3^{n/3})$ and polynomial space. The approach is based on finding a shortest rural postman path in the de Bruijn graph of the given set of strings. The algorithm works basically by exhaustive search, but having reduced the search space to size $3^{n/3}$, and then inspecting each possibility in polynomial time. We show that for the case of 3-strings to find an optimal rural path it is enough to guess where such a path enters each weakly connected component formed by input strings. We then show that for a component on k arcs there are at most k such entry points. Since the total number of arcs is n, the running time is roughly $k^{n/k}$ and this does not exceed $3^{n/3}$ (for $k \in \mathbb{N}$).

The current situation with exact algorithms for SCS is similar to what is known for some other NP-hard problems — say, the satisfiability problem (SAT), the maximum satisfiability problem (MAX-SAT), and the traveling salesman problem (TSP). Namely, despite many efforts the best known algorithms for the general versions of these problems run in time $O^*(2^n)$ (n being the number of variables/vertices). At the same time better upper bounds are known for special cases of these problems: $O(1.308^n)$ for 3-SAT [13], $O(1.731^n)$ for MAX-2-SAT [28], $O(1.251^n)$ for TSP on cubic graphs [14], $O^*(1.109^n)$ for $(n, 3)$-MAX-2-SAT [19], c^n (where $c < 2$) for SAT [5] and MAX-SAT [8,19] on formulas with constant clause density. Moreover, it is known that k-SAT can be solved in time $O((2 - 2/k)^n)$ [22] and TSP can be solved in time $O((2 - \epsilon)^n)$, where $\epsilon > 0$ depends only on the degree bound of a graph [4].

2 General setting

Throughout the paper $\mathcal{S} = \{s_1, \ldots, s_n\}$ is an input set of strings over an alphabet Σ, and n is the number of strings. W.l.o.g. we assume that no s_i is a substring of s_j for any $i \neq j$.

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

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

2.1 Suffix graphs and the traveling salesman problem

Clearly, $s \circ \mathrm{suffix}(s,t)$ is the shortest string containing s and t in this order. More generally, the shortest string containing strings $s_{i_1}, \ldots, s_{i_n}$ in this order is

$$s_{i_1} \circ \mathrm{suffix}(s_{i_1}, s_{i_2}) \circ \cdots \circ \mathrm{suffix}(s_{i_{n-1}}, s_{i_n})\,.$$

Thus, the goal of SCS is to find a permutation of n input strings minimizing the total length of the suffix function. As mentioned in the introduction, one can define a complete directed graph on the given set $\{s_1, \ldots, s_n\}$ of n strings as a set of vertices where vertices s_i and s_j are joined by an arc of weight $|\mathrm{suffix}(s_i, s_j)|$ (suffix graph). Solving SCS then corresponds to solving TSP in this graph. This connection has been used in essentially all previous approximation algorithms for SCS. This graph however is asymmetric (i.e., directed) and the best known algorithm due to Bellman [2], Held and Karp [12] uses $O^*(2^n)$ time and space. There are also algorithms based on inclusion-exclusion with running time $O^*(2^n \cdot M)$ and space $O^*(M)$ [18,16,1] (here, M is the maximal arc weight). Lokshtanov and Nederlof [21] show how to solve TSP in $O^*(2^n \cdot M)$ time with only $O^*(1) = \mathrm{poly}(n, \log M)$ space. For symmetric TSP, Björklund [3] recently came up with an $O^*(1.657^n \cdot M)$ time randomized algorithm. Note that for SCS, M does not exceed the size of the input, hence the mentioned inclusion-exclusion algorithm solves SCS in $O^*(2^n)$ time and polynomial space.

2.2 De Bruijn graphs and the rural postman problem

In this paper we deal with another useful concept, namely *de Bruijn graphs*. Such graphs are widely used in genome assembly, one of the most important practical applications of SCS [25]. At the same time they have only few applications in theoretical investigations of SCS. To simplify its definition from now on we stick to strings of length 3 only. So, let $\mathcal{S} = \{s_1, \ldots, s_n\}$ be a set of 3-strings over the alphabet Σ. The de Bruijn graph DG is a weighted complete directed graph (with loops, but without multiple arcs) with the set of vertices Σ^2. Distinct vertices s and t are joined by an arc of weight $|\mathrm{suffix}(s,t)|$. Also, for each string from Σ^2 consisting of the same two symbols there is a loop of weight 1. Intuitively, the weight of an arc (s,t) is equal to the number of symbols we need to spell going from a string s to a string t. (Particularly, going from a string `AA` to itself we need to spell one more `A`, that is why loops are of weight 1.) Thus, all arcs in DG have weight either 1 or 2. Note that any 3-string s over Σ defines an arc of weight 1 in DG: the arc joins the prefix of s of length 2 and the suffix of s of

length 2. Thus, what we are looking for in the SCS problem is a shortest path in DG going through all the arcs $E_{\mathcal{S}}$ given by $\mathcal{S}$.

This problem is known as *directed rural postman path problem* (DRPP). In DRPP one is given a weighted graph $G = (V, E)$ and a subset $E_R \subseteq E$ of its arcs and the goal is to find a shortest path in G going through all the arcs of E_R. The arcs from E_R are called *required*, all the remaining arcs are called *optional*. A path going through all the required arcs is called a *rural path* and a shortest such path is called an *optimal rural path*.[5]

DRPP has many practical applications (see, e.g., [9,11]). At the same time almost no non-trivial exact algorithms are known for DRPP. As with SCS and TSP, for DRPP there is a simple algorithm with the running time $O^*(n!)$ (for DRPP, by n we denote the number of required arcs) as well as dynamic programming based algorithm with running time $O^*(2^n)$. DRPP generalizes such problems as Chinese Postman Problem and Asymmetric TSP. If the set of required arcs forms a single weakly connected component (weakly connected components of a directed graph are just connected components in this graph with all directed arcs replaced by undirected edges), then the problem can be solved in polynomial time: all one needs to do is to add arcs of minimal total weight to imbalanced vertices. This can be done by finding a minimum weight perfect matching in an appropriate bipartite graph (details can be found, e.g., in [6]). If the set of required arcs forms more than just one weakly connected component then DRPP becomes NP-hard [20]. The reason is that now one not only needs to balance all the imbalanced vertices by adding arcs of minimal total weight but also to guarantee somehow that the resulting graph is connected. This turns out to be harder.

However 2-SCS can be solved in polynomial time even if the input strings form more than one weakly connected component. The important property of 2-SCS (as opposed to, say, 3-SCS) is that 2-strings from different weakly connected components always have zero overlap. This means that one can find an optimal rural path for each component separately.

3 Algorithm

The set $\mathcal{S}$ of 3-strings defines the required set of arcs $E_{\mathcal{S}}$ in the de Bruijn graph DG. What we are looking for is a shortest path in this graph going through all the required arcs (an optimal rural path). Optional arcs have weight 1 or 2 while all required arcs have weight 1. For each vertex of the graph we know the number of adjacent incoming and outgoing required arcs, but we do not know the number of adjacent incoming and outgoing optional arcs (in an optimal rural path).

Note the following two simple properties of an optimal rural path.

[5] We use the term "path" to denote a path that may go through some vertices and arcs more than once (a term "walk" is also used in the literature for this). A simple path is a path without repeated vertices and arcs.

– An optimal rural path does not start and does not end with an optional arc (removing such an arc leaves a rural path of smaller weight).
– There always exists an optimal rural path that does not contain an optional arc followed by another optional arc. Two such arcs can be replaced by a single arc. Since all arcs have weight 1 or 2 this does not increase the total weight of a path.

By $d^{\mathrm{re}}_{\mathrm{in}}(v)$ and $d^{\mathrm{re}}_{\mathrm{out}}(v)$ we denote the number of required incoming and outgoing arcs to v, respectively: $d^{\mathrm{re}}_{\mathrm{in}}(v) = |\{(u,v) \in E_{\mathcal{S}}\}|$ and $d^{\mathrm{re}}_{\mathrm{out}}(v) = |\{(v,w) \in E_{\mathcal{S}}\}|$. Similarly, for a path P in DG, by $d^{\mathrm{op}}_{\mathrm{in}}(P,v)$ and $d^{\mathrm{op}}_{\mathrm{out}}(P,v)$ we denote the number of optional incoming and outgoing arcs for the vertex v in the path P:

$$d^{\mathrm{op}}_{\mathrm{in}}(P,v) = |\{(u,v) \mid (u,v) \in P, (u,v) \notin E_{\mathcal{S}}\}|,$$
$$d^{\mathrm{op}}_{\mathrm{out}}(P,v) = |\{(v,w) \mid (v,w) \in P, (v,w) \notin E_{\mathcal{S}}\}|\,.$$

Recall that a path may go through a particular vertex more than once, so these degrees may be greater than 1. Also, the path P may go through a particular arc more than once hence the sets in the right hand side of the definition $d^{\mathrm{op}}_{\mathrm{in}}(P,v)$ and $d^{\mathrm{op}}_{\mathrm{out}}(P,v)$ are actually multisets. In other words, $d^{\mathrm{op}}_{\mathrm{in}}(P,v)$ ($d^{\mathrm{op}}_{\mathrm{out}}(P,v)$) is the number of times the path P enters (respectively, leaves) the vertex v by an optional arc.

Definition 1 (configuration). *A* configuration *is a pair $f = (f_{in}, f_{out})$ of functions from V to $\mathbb{N}$. A configuration tells for each vertex the number of incoming and outgoing optional arcs of a path. Consequently we say that a configuration* is consistent *with a path P iff $f_{in}(v) = d^{op}_{in}(P,v)$ and $f_{out}(v) = d^{op}_{out}(P,v)$ for each vertex v. A path in DG determines a configuration in a natural way.*

Definition 2 (normal configuration, special vertex). *We say that a configuration $f = (f_{in}, f_{out})$ is* normal *iff the following three conditions hold.*

– *It is consistent with at least one rural path. This, in particular, means that for all but two vertices v (the two exceptional vertices being the first and the last vertices of a path)*

$$f_{in}(v) + d^{re}_{in}(v) = f_{out}(v) + d^{re}_{out}(v). \tag{1}$$

– *For each weakly connected component $\mathcal{C}$ of $E_{\mathcal{S}}$,*

$$\sum_{v \in \mathcal{C}} \min\{f_{in}(v), f_{out}(v)\} \leq 1\,. \tag{2}$$

I.e., each weakly connected component contains at most one vertex that has both incoming and outgoing optional arcs. Moreover if such a vertex exists then it has just one incoming or one outgoing arc. Such a vertex is called special.

– *For each vertex v, if v has only incoming (outgoing) required arcs then it has only outgoing (incoming) optional arcs:*

$$(d^{re}_{out}(v) = 0 \Rightarrow f_{in}(v) = 0) \text{ and } (d^{re}_{in}(v) = 0 \Rightarrow f_{out}(v) = 0). \tag{3}$$

In particular, a vertex v with $\min\{d^{re}_{in}(v), d^{re}_{out}(v)\} = 0$ cannot be special.

Definition 3 (normal path). *A normal path is a path with a normal configuration.*

The motivation for studying configurations is given by the following lemmas (which are proven below).

Lemma 1. *There exists an optimal rural path that is normal.*

Lemma 2. *Given a normal configuration f of an (unknown) optimal rural path we can find in polynomial time an optimal rural path consistent with f.*

It remains to show that the number of different normal configurations is not too large. This is guaranteed by the following lemmas.

Lemma 3. *A weakly connected component $\mathcal{C}$ of E_S consisting of k arcs has at most k different normal configurations.*

Lemma 4. *All normal configurations can be enumerated in time $O^*(3^{n/3})$ and polynomial space.*

Using these four lemmas the main result of the paper follows almost immediately.

Theorem 1. *The 3-SCS problem can be solved in time $O^*(3^{n/3})$ and polynomial space.*

Proof. Due to Lemma 4 we can enumerate all normal configurations in time $O^*(3^{n/3})$ and polynomial space. By Lemma 1 at least one of these configurations corresponds to an optimal rural path. Given such a configuration we can recover an optimal rural path by Lemma 2. □

3.1 Proofs

In this subsection, we complete the analysis of the algorithm by proving the lemmas given in the previous subsection. In the proofs, we often consider a path as a sequence of vertices. In this notation, lower case letters are used to denote vertices while upper case letters denote parts of a path, i.e., sequences of vertices (possibly empty). E.g., to specify that a path P starts with a vertex s, goes through a vertex v and ends in a vertex t we write $P = sAvBt$. In the pictures below, required arcs are shown in bold, optional arcs are thin and gray, snaked arcs denote just a part of a path.

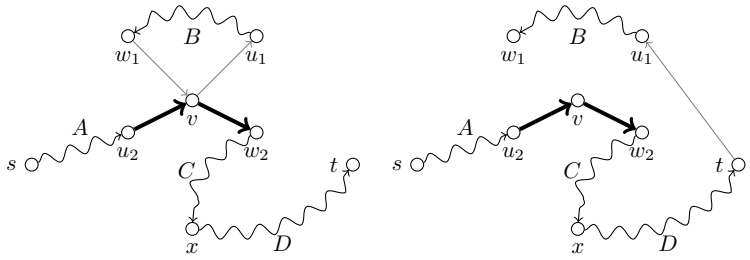
Proof (of Lemma 1). Let P be an optimal rural path containing the minimal number of optional arcs. We show that if P is not a normal path, then the number of optional arcs in P can be decreased without increasing its weight. This is done by replacing two optional arcs with a new one. Since all arcs have weights 1 or 2, this replacement does not increase the weight of P.

Consider a weakly connected component $\mathcal{C}$ and let x be the last vertex of $\mathcal{C}$ in the path P. To guarantee that (2) holds we first transform P such that for all vertices v of $\mathcal{C}$ with the only possible exception of x we have

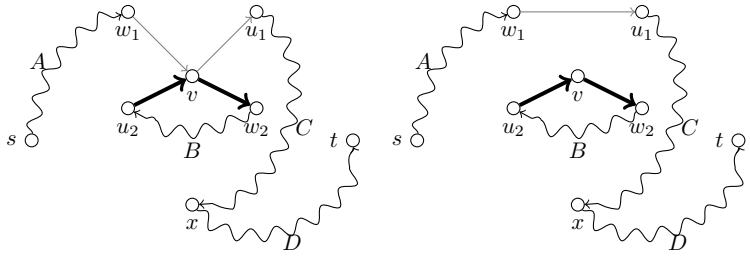
$$\min\{d^{\mathrm{op}}_{\mathrm{in}}(P,v), d^{\mathrm{op}}_{\mathrm{out}}(P,v)\} = 0\,.$$

Assume that a vertex $v \neq x$ not fulfilling this equality exists in $\mathcal{C}$. Denote incoming and outgoing optional arcs of v by (w_1, v) and (v, u_1), respectively. Let u_2 be a vertex such that the arc (u_2, v) precedes the arc (v, u_1) in P and w_2 be a vertex such that the arc (v, w_2) follows the arc (w_1, v) in P. Since the path does not contain two consecutive optional arcs, the arcs (u_2, v) and (v, w_2) differ from the arcs (w_1, v) and (v, u_1). We now consider the following two cases depending on whether the path first goes through (u_2, v) and (v, u_1) or through (w_1, v) and (v, w_2).

1. The path P has the form $sAu_2vu_1Bw_1vw_2CxDt$ (i.e., P first goes through (u_2, v) and (v, u_1) and only then through (w_1, v) and (v, w_2)). We transform it to $sAu_2vw_2CxDtu_1Bw_1$. Note that this transformation increases the number of optional arcs out of t, but it reduces the total number of optional arcs.



2. The path P has the form $sAw_1vw_2Bu_2vu_1CxDt$. We then replace the arcs (w_1, v) and (v, u_1) by a new arc (w_1, u_1). As a result we get the path sAw_1u_1CxDt and a cycle vw_2Bu_2v. Recall however that $\mathcal{C}$ is a weakly connected component. This means that the new path has at least one vertex in common with the cycle. Thus we can glue this cycle into this path.



Clearly both transformations above do not break the path and decrease the total number of optional arcs.

We now show that P can be transformed so that $\min\{d^{\mathrm{op}}_{\mathrm{in}}(P,x), d^{\mathrm{op}}_{\mathrm{out}}(P,x)\} \le 1$. Assume for the sake of contradiction that x has in P at least two incoming and at least two outgoing optional arcs. Let (v,x) and (x,w) be the first optional incoming and outgoing arcs for x in P. Consider two subcases.

1. P first goes through (v,x) and then through (x,w). Since P has at least two optional arcs out of x the path P has the form $sAvxBxwCxDt$. We transform it to $sAvwCxBxDt$.
2. P first goes through (x,w). Then it has the form $sAxwBvxCt$ and can be transformed to $sAxCtwBv$.

Thus, P satisfies (2).

Finally, we show how to transform P so that (3) holds. Consider a vertex $v \in \mathcal{C}$ and assume w.l.o.g. that it has no incoming required arcs (i.e., $d^{\mathrm{re}}_{\mathrm{in}}(v) = 0$). Assume that P also has an optional arc (v,w). Since P cannot start with an optional arc it has an arc (u,v) preceding (v,w) and this arc is also optional. But then two optional arcs (u,v) and (v,w) can be replaced with an arc (u,w). This again contradicts the assumption that P has the minimal possible number of optional arcs. The case $d^{\mathrm{re}}_{\mathrm{out}}(v) = 0$ is treated similarly. Thus, P satisfies (3).

We conclude that any rural path with the minimal number of optional arcs satisfies the properties (2) and (3). The property (1) holds for such a path for a trivial reason. Thus, any such path is normal. □

Proof (of Lemma 2). In the following we assume that we know the first vertex s and the last vertex t of an optimal rural path that we are looking for. Since the first and the last arc of such a path are both required arcs, enumerating all such pairs (s,t) can be done in $O(n^2)$ time.

To find the required path we modify the graph DG and the set of required arcs $E_{\mathcal{S}}$ as follows:

- Introduce $|\Sigma|$ new vertices labeled by single symbols and join them to all other vertices by arcs of weight equal to the length of the suffix of the two corresponding strings. E.g., $w(\mathtt{A},\mathtt{AB}) = 1$, $w(\mathtt{A},\mathtt{BC}) = 2$, $w(\mathtt{BC},\mathtt{A}) = 1$, $w(\mathtt{BA},\mathtt{A}) = 0$, $w(\mathtt{A},\mathtt{B}) = 1$.
- For each vertex v of the initial graph DG labeled by $\mathtt{AB}$ add $f_{\mathrm{in}}(v)$ copies of the arc $(\mathtt{A},\mathtt{AB})$ and $f_{\mathrm{out}}(v)$ copies of $(\mathtt{AB},\mathtt{B})$ to the set of required arcs $E_{\mathcal{S}}$.

Denote the resulting graph by DG' and the resulting set of required arcs by $E'_{\mathcal{S}}$. It is worth to note that $E'_{\mathcal{S}}$ is a multiset, namely it might contain several copies of new required arcs (e.g., $f_{\mathrm{in}}(\mathtt{AB})$ copies of the arc $(\mathtt{A},\mathtt{AB})$).

Let $C_1,\dots,C_p$ be the weakly connected components of $E_{\mathcal{S}}$ and $C'_1,\dots,C'_q$ be the weakly connected components of $E'_{\mathcal{S}}$. Clearly $q \le p$ and for each C_i there is C'_j such that $C_i \subseteq C'_j$.

First we show that the weight of an optimal rural path with configuraion f in DG is equal to the weight of an optimal rural path in DG'. Indeed, given an optimal rural path P consistent with f in DG one replaces each its optional arc $(\mathtt{AB},\mathtt{CD})$ (of weight 2) with three arcs $(\mathtt{AB},\mathtt{B})$, $(\mathtt{B},\mathtt{C})$, $(\mathtt{C},\mathtt{CD})$ (of total weight $0+1+1=2$) and each optional arc $(\mathtt{AB},\mathtt{BC})$ (of weight 1) with two arcs $(\mathtt{AB},\mathtt{B})$,

$(\mathtt{B}, \mathtt{BC})$ (of total weight $0+1=1$). The resulting path P' is a rural path in DG': we replaced exactly $f_{\text{out}}(\mathtt{AB})$ optional arcs out of the vertex $\mathtt{AB}$ with new required arcs $(\mathtt{AB}, \mathtt{B})$. Moreover, this path clearly has exactly the same weight. Conversely, let P' be an optimal rural path in DG'. Just by removing all vertices labeled by single symbols we get a rural path P consistent with f whose weight is not greater than the weight of P'.

Now we show that an optimal rural path in DG' can be found in polynomial time. For this, we show that it is enough to solve the problem for each weakly connected component of $E'_{\mathcal{S}}$ separately.

Let P' be such an optimal rural path in DG'. Translate it back to a path P in DG by removing all vertices labeled by single symbols. Let $P = A_1 A_2 \dots A_k$ where each sequence of vertices A_i lies inside the same weakly connected component C'_j of $E'_{\mathcal{S}}$ and A_i and A_{i+1} belong to different components. Denote by u_i, v_i the first and the last vertex of A_i (recall that the path does not contain an optional arc followed by another optional arc). A simple but crucial observation is that each arc (v_i, u_{i+1}) has weight 2. Indeed, if $w(v_i, u_{i+1}) = 1$ then $v_i = \mathtt{AB}$ and $u_{i+1} = \mathtt{BC}$. Note that (v_i, u_{i+1}) is an optional arc since v_i and u_{i+1} belong to different components of $E'_{\mathcal{S}}$ (and hence to different components of $E_{\mathcal{S}}$). This means that $f_{\text{out}}(v_i) > 0$ and $f_{\text{in}}(u_{i+1}) > 0$. But then the arcs $(\mathtt{AB}, \mathtt{B})$ and $(\mathtt{B}, \mathtt{BC})$ are required in DG' and thus v_i and u_{i+1} lie in the same weakly connected component of $E'_{\mathcal{S}}$.

We would like to show now that there exists an optimal rural path P' in DG' that goes through each component of $E'_{\mathcal{S}}$ separately. For this, we show that if P' enters the same component of $E'_{\mathcal{S}}$ more than once then we can reduce the number of optional arcs between the components by transforming a path (without increasing the total weight of the path). As before, translate the path P' back to P. Now assume that for some component C'_j, the path P enters C'_j at least two times, i.e., there are two optional arcs (a_1, b_1) and (a_2, b_2) in P such that $b_1, b_2 \in C'_j$ and $a_1, a_2 \notin C'_j$. Assume that C'_j is not the last component of the path P (the case when it is the last one is similar). This means that P must also leave the component C'_j two times. More formally, P contains two optional arcs (b_3, a_3) and (b_4, a_4) where $b_3, b_4 \in C'_j$ and $a_3, a_4 \notin C'_j$. Replace now the arcs (a_1, b_1) and (b_3, a_3) by (b_3, b_1) and (a_1, a_3). It is easy to see that such a transformation does not change the degrees of vertices. To guarantee that the resulting set of arcs is a single path but not a cycle and a path we note that b_1, b_2, b_3, b_4 lie in the same weakly connected component. Also, the weight of the path is not increased (since $w(a_1, b_1) = w(b_3, a_3) = 2$ while $w(b_3, b_1), w(a_1, a_3) \leq 2$).

Thus, to find an optimal rural path in DG' we can find an optimal path for each component of $E'_{\mathcal{S}}$ separately and then join the found paths arbitrarily (recall that solving DRPP for a weakly connected component is a polynomial problem). □

Proof (of Lemma 3). Let

$$\text{mindeg}^{\text{re}}(v) = \min\{d^{\text{re}}_{\text{in}}(v), d^{\text{re}}_{\text{out}}(v)\},\ \text{mindeg}^{\text{op}}(v) = \min\{f_{\text{in}}(v), f_{\text{out}}(v)\}.$$

By definition of a normal configuration (see (3)) each component contains at most one special vertex, i.e., a vertex with $\mathrm{mindeg}^{\mathrm{op}} = 1$. Recall from the proof of Lemma 4 that we only need to know which vertex in a configuration is special (if any) to fully determine the configuration.

We now consider the following two cases.

$\mathcal{C}$ is Eulerian. Clearly $\mathcal{C}$ contains at most k vertices (and contains exactly k vertices when it is a simple cycle). Note that if $E_{\mathcal{S}}$ does not consist of $\mathcal{C}$ only then $\mathcal{C}$ must contain at least one special vertex in any rural path and hence the number of different normal configurations for $\mathcal{C}$ is k. At the same time, if $E_{\mathcal{S}}$ contains $\mathcal{C}$ only then an optimal rural path can be found in polynomial time.

$\mathcal{C}$ is not Eulerian. By (3), it is enough to show that $\mathcal{C}$ contains at most $(k-1)$ vertices with non-zero $\mathrm{mindeg}^{\mathrm{re}}$. Then either one of these $(k-1)$ vertices is special or there are no special vertices — thus, at most k different configurations.

To show that there are at most $(k-1)$ vertices in $\mathcal{C}$ with non-zero $\mathrm{mindeg}^{\mathrm{re}}$ consider two subcases.

1. By removing directions of the arcs in $\mathcal{C}$ we get a simple path on k arcs. Then $\mathcal{C}$ contains $(k+1)$ vertices but both ends of this path have zero $\mathrm{mindeg}^{\mathrm{re}}$.
2. Otherwise $\mathcal{C}$ contains at most k vertices. If the number of vertices is strictly smaller than k then we are done. If the number of vertices is equal to k we find a vertex with zero $\mathrm{mindeg}^{\mathrm{re}}$. For this, take any vertex in $\mathcal{C}$ and start a path from it. As a result we either arrive to a vertex with zero out-degree (in this case we are done) or construct a cycle. Since $\mathcal{C}$ is weakly connected for at least one of the vertices of this cycle the sum of in-degree and out-degree is at least 3. But then $\mathcal{C}$ must contain a vertex with in-degree plus out-degree equal to 1 and we are done again.

□

Proof (of Lemma 4). Let $E_{\mathcal{S}}$ consist of t weakly connected components $\mathcal{C}_1, \ldots, \mathcal{C}_t$, let also n_i be the number of required arcs in $\mathcal{C}_i$ (hence $n_1 + \cdots + n_t = n$). By Lemma 3 above, for $\mathcal{C}_i$ there are at most n_i different configurations. Thus, the total number of normal configurations for $E_{\mathcal{S}}$ is at most $\prod_{i=1}^{t} n_i$. We show that this is at most $3^{n/3}$ by induction on n. The base case $n = 1$ is clear. Induction step:

$$\prod_{i=1}^{t} n_i = n_t \cdot \prod_{i=1}^{t-1} n_i \leq 3^{\frac{n-n_t}{3}} n_t = 3^{\frac{n-n_t}{3} + \log_3 n_t}.$$

This does not exceed $3^{n/3}$ since $\log_3 n_t \leq n_t/3$ for any $n_t \in \mathbb{N}$.

Enumerating all normal configurations is easy: for each weakly connected component we just need to select a special vertex. Indeed, if a vertex $v \neq s, t$ is special then $\min\{f_{\mathrm{in}}(v), f_{\mathrm{out}}(v)\} = 1$, otherwise $\min\{f_{\mathrm{in}}(v), f_{\mathrm{out}}(v)\} = 0$. The exact values of $f_{\mathrm{in}}(v)$ and $f_{\mathrm{out}}(v)$ can be then derived from the equality (1). □

4 Further directions

The natural open question is to solve SCS in less than 2^n steps. An apparently easier problem is to prove an upper bound $O^*(2^{\alpha(r)n})$ for r-SCS where $\alpha(r) < 1$ for all r.

Acknowledgments

Research is partially supported by Russian Foundation for Basic Research (12-01-31057-mol_a), 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) and Computer Science Club scholarship.

Also, we would like to thank the anonymous reviewers for many valuable comments that helped us to improve the readability of the paper.

References

1. Bax, E., Franklin, J.: A Finite-Difference Sieve to Count Paths and Cycles by Length. Inf. Process. Lett. 60, 171–176 (November 1996)
2. Bellman, R.: Dynamic Programming Treatment of the Travelling Salesman Problem. J. ACM 9, 61–63 (January 1962)
3. Björklund, A.: Determinant Sums for Undirected Hamiltonicity. In: Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. pp. 173–182. FOCS'10, IEEE Computer Society, Washington, DC, USA (2010)
4. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: The traveling salesman problem in bounded degree graphs. ACM Trans. Algorithms 8(2), 18:1–18:13 (Apr 2012)
5. Calabro, C., Impagliazzo, R., Paturi, R.: A Duality between Clause Width and Clause Density for SAT. In: Proceedings of the 21st Annual IEEE Conference on Computational Complexity. pp. 252–260. CCC '06, IEEE Computer Society (2006)
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. Dantsin, E., Wolpert, A.: 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. LNCS, vol. 4121, pp. 266–276 (2006)
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. Held, M., Karp, R.M.: The Traveling-Salesman Problem and Minimum Spanning Trees. Mathematical Programming 1, 6–25 (1971)
13. Hertli, T.: 3-SAT Faster and Simpler - Unique-SAT Bounds for PPSZ Hold in General. In: Foundations of Computer Science (FOCS). pp. 277 –284 (oct 2011)
14. Iwama, K., Nakashima, T.: An Improved Exact Algorithm for Cubic Graph TSP. In: Computing and Combinatorics, LNCS, vol. 4598, pp. 108–117. Springer Berlin / Heidelberg (2007)
15. 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)
16. Karp, R.M.: Dynamic Programming Meets the Principle of Inclusion and Exclusion. Operations Research Letters 1(2), 49–51 (1982)
17. Karpinski, M., Schmied, R.: Improved Lower Bounds for the Shortest Superstring and Related Problems. CoRR abs/1111.5442 (2011)
18. Kohn, S., Gottlieb, A., Kohn, M.: A Generating Function Approach to the Traveling Salesman Problem. In: ACN'77: Proceedings of the 1977 annual conference. pp. 294–300. New York, NY, USA (1977)
19. Kulikov, A., Kutzkov, K.: New upper bounds for the problem of maximal satisfiability. Discrete Mathematics and Applications 19, 155–172 (2009)
20. Lenstra, J.K., Kan, A.H.G.R.: Complexity of vehicle routing and scheduling problems. Networks 11(2), 221–227 (1981)
21. Lokshtanov, D., Nederlof, J.: Saving space by algebraization. In: Proceedings of the 42nd ACM symposium on Theory of computing. pp. 321–330. STOC '10, ACM (2010)
22. Moser, R.A., Scheder, D.: A full derandomization of Schöning's k-SAT algorithm. In: Proceedings of the 43rd annual ACM symposium on Theory of computing. pp. 245–252. STOC '11, ACM (2011)
23. 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), to appear
24. 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)
25. 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)
26. Sweedyk, Z.: $2\frac{1}{2}$-Approximation Algorithm for Shortest Superstring. SIAM J. Comput. 29(3), 954–986 (Dec 1999)
27. 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)
28. Williams, R.: A new algorithm for optimal 2-constraint satisfaction and its implications. Theoretical Computer Science 348(2-3), 357–365 (2005)