

# Rotations of Periodic Strings and Short Superstrings

Dany Breslauer

Max-Planck-Institute für Informatik

June 24, 1996

[summary by Mireille Régnier]

## 1. State of the Art

Let  $S = \{s_1, \dots, s_m\}$  be a set of strings over some alphabet  $\Sigma$ . A *common superstring*, or simply *superstring*, of  $S$  is a string  $s$  such that each  $s_i$  in  $S$  is a substring (*i.e.*, a consecutive block) of  $s$ . The shortest superstring problem is to find a superstring of the smallest possible length for any given set of strings  $S$ . The problem has applications in a wide range of areas including data compression [6] and DNA sequencing.

Since the problem is *NP-hard* [6] a lot of effort has been taken to find good approximation algorithms with guaranteed performance. Blum *et al.* [4] showed that the problem is *MAX SNP-hard* and thus does not have a polynomial time approximation scheme unless  $P = NP$ . Tarhio and Ukkonen [9] and Turner [11] gave several approximation algorithms that achieve  $\frac{1}{2}$ -approximation with respect to the *compression* measure, or the *total overlap* between adjacent strings in a superstring. This approximation ratio has been improved to  $\frac{38}{63}$  by Kosaraju *et al.* [7]. Notice that superstrings have the minimum length if and only if they induce the maximum total overlap. Such relation, however, does not hold for approximations, and a good approximation for the length of the shortest superstring is not necessarily a good approximation for the maximum overlap in the superstring, and *vice versa*.

The first constant-approximation algorithm for the length of the shortest superstring was given by Blum *et al.* [4], who discovered a 3-approximation algorithm and proved that the “Greedy” algorithm by Tarhio and Ukkonen [9] achieves 4-approximation. Their algorithms and analysis rely on the close relation between the shortest superstring problem, that was shown by Turner [11] to be reducible to the *travelling salesman* problem, and the *cycle cover* problem. The same relation was exploited in subsequent papers [10] ( $\approx 2.89$ ), [5] ( $\approx 2.83$ ), [7] ( $\approx 2.79$ ) and [1, 2] ( $\approx 2.75$ ). Armen and Stein [3] have also recently obtained a  $2\frac{2}{3}$ -approximation algorithm, independently of our work.

Here we continue this line of work, and further improve the approximation ratio to  $2\frac{2}{3} \approx 2.67$  and to  $2\frac{25}{42} \approx 2.596$ . The improved algorithms are similar to the previous algorithms in the sense that they construct a superstring by computing some optimal cycle covers on the *distance graph* of the given input strings, and then break and merge the cycles to finally obtain a Hamiltonian path representing some superstring. The key to the improvement are new bounds on the overlap between two strings.

## 2. Preliminaries

Without loss of generality, we assume that the set  $S$  is “substring-free” in that no string  $s_i \in S$  is a substring of any other  $s_j \in S$ . For two strings  $s$  and  $t$ , let  $y$  be the longest string such that  $s = xy$

and  $t = yz$  for some *non-empty* strings  $x$  and  $z$ . We denote  $\text{ov}(s, t) = |y|$  the *overlap* between  $s$  and  $t$ ,  $d(s, t) = |x|$  the *distance* from  $s$  to  $t$  and  $\text{pref}(s, t) = x$ . Given a list of strings  $s_{i_1}, \dots, s_{i_r}$ , we define the superstring  $s = \langle s_{i_1}, \dots, s_{i_r} \rangle$  to be  $\text{pref}(s_{i_1}, s_{i_2}) \text{pref}(s_{i_2}, s_{i_3}) \cdots \text{pref}(s_{i_{r-1}}, s_{i_r}) s_{i_r}$ . It is clear that each shortest superstring for  $S$  must be  $\langle s_{i_1}, \dots, s_{i_m} \rangle$  for some permutation  $i_1, \dots, i_m$  of  $\{1, \dots, m\}$ . Its length,  $\text{opt}(S)$ , and the total overlap between adjacent strings,  $\text{maxov}(S)$ , satisfy:  $\text{opt}(S) = \sum_{s_i \in S} |s_i| - \text{maxov}(S)$ .

**2.1. Distance graph and cycle covers.** The concept of a *distance graph* is central to all existing approximation algorithms for shortest superstrings. Let  $G_S = (V, E, w)$  be a directed graph, where the set of vertices  $V = \{s_1, \dots, s_m\}$ , the set of edges  $E = \{(s_i, s_j) \mid 1 \leq i \neq j \leq m\}$ , and the weight function  $w$  is the distance function  $d(, )$ .  $G_S$  is called the distance graph of  $S$ . If we denote the cost of a minimum weight Hamiltonian cycle on  $G_S$  as  $\text{Tsp}(G_S)$ , then obviously, for any  $s_i \in S$ ,

$$\text{Tsp}(G_S) \leq \text{opt}(S) \leq \text{Tsp}(G_S) + |s_i|.$$

In other words, a minimum weight Hamiltonian cycle on  $G_S$  would be a very good approximation of a shortest superstring of  $S$ . Since TSP is NP-hard and has no good approximation algorithms, we try to work with a relaxed version of TSP, the *cycle cover* problem defined below.

Given a directed weighted graph  $G$ , a *cycle cover* is a set of (simple) cycles such that each vertex is contained in exactly one cycle. The weight of the cycle cover is the total weight of its cycles. A minimum weight cycle cover can be computed in  $O(n^3)$  time using the Hungarian algorithm [8].

Let  $\text{Cyc}(G_S)$  be the weight of a minimum weight cycle cover of  $G_S$ . Then we have  $\text{Cyc}(G_S) \leq \text{Tsp}(G_S) \leq \text{opt}(S)$ . To get an upper bound on  $\text{opt}(S)$  in terms of  $\text{Cyc}(G_S)$  we have to look at the particular structures and properties of strings.

**2.2. Periodicity of strings and semi-infinite strings.** A string  $x$  is a *factor* of a string  $s$  if  $s = x^i y$  for some positive integer  $i$  and prefix  $y$  of  $x$  ( $y$  may be empty). The *factor* of a non-empty string  $s$ , denoted  $\text{factor}(s)$ , is the *shortest* factor of  $s$  and the *period* of  $s$  is denoted  $\text{period}(s) = |\text{factor}(s)|$ . A semi-infinite string  $s = a_1 a_2 \cdots$  is said to be *periodic* if  $s = xs$  for some *non-empty* string  $x$ . The shortest such  $x$  is called the factor of  $s$ . Two (periodic semi-infinite) strings  $s, t$  are *equivalent* if their factors are cyclic shifts of each other, *i.e.*, if there are strings  $x, y$  such that  $\text{factor}(s) = xy$  and  $\text{factor}(t) = yx$ . Otherwise, they are *inequivalent*. For each string  $s$ , let  $s^\infty$  denote the semi-infinite string  $ss \cdots$ , and  $s_\infty = \text{factor}(s)^\infty$  denote the periodic semi-infinite string that is equivalent to  $s$  and begins with  $s$ . Note that in general  $s^\infty \neq s_\infty$ . For example,  $(010)^\infty = 010010 \cdots \neq (010)_\infty = 0101 \cdots$ .

Connections between a cycle in  $G_S$  and the periodicity of the strings obtained by breaking the cycle are essentially given in [4]. Let  $c = s_{i_1}, \dots, s_{i_r}, s_{i_1}$  be a cycle in  $G_S$ , and  $w(c)$  be its weight. Without loss of generality, assume that  $c$  has the minimum weight among all cycles in  $G_S$  containing  $s_{i_1}, \dots, s_{i_r}$ . We will use:

LEMMA 1.  $w(c) = d(s_{i_1}, s_{i_2}) + \cdots + d(s_{i_{r-1}}, s_{i_r}) + d(s_{i_r}, s_{i_1}) = \text{period}(\langle s_{i_1}, \dots, s_{i_r} \rangle)$ .

**2.3. The overlap-rotation lemma.** The key to the improved approximation bounds is our overlap-rotation lemma below that follows from the classical *Critical Factorization Theorem*. Given a semi-infinite string  $\alpha = a_1 a_2 \cdots$ , we denote the rotation  $\alpha[k] = a_k a_{k+1} \cdots$ .

LEMMA 2. *Let  $\alpha$  be a periodic semi-infinite string. There exists an integer  $k$ , such that for any (finite) string  $s$  that is inequivalent to  $\alpha$ ,*

$$\text{ov}(s, \alpha[k]) < \text{period}(s) + \frac{1}{2} \text{period}(\alpha).$$

- (1) Construct the distance graph  $G_S$  for set  $S$ .
- (2) Find a minimum weight cycle cover  $C$  on the graph  $G_S$ .
- (3) For each cycle  $c = s_{i_1}, \dots, s_{i_r}, s_{i_1} \in C$ , choose a string  $t_c$  such that for some  $j$ ,  $t_c$  contains  $\langle s_{i_{j+1}}, \dots, s_{i_r}, s_{i_1}, \dots, s_{i_j} \rangle$ , and  $t_c$  is contained in  $\langle s_{i_j}, \dots, s_{i_r}, s_{i_1}, \dots, s_{i_{j-1}}, s_{i_j} \rangle$ .
- (4) Let  $T$  be the set of all strings chosen above and construct the distance graph  $G_T$  for  $T$ .
- (5) Find a minimum weight cycle cover  $CC$  on  $G_T$ .
- (6) Break each cycle of  $CC$  arbitrarily to obtain a superstring of the elements in the cycle.
- (7) Concatenate the strings found at Step (6) arbitrarily to produce a superstring  $\tilde{s}$  of  $S$ .

FIGURE 1. The generic shortest superstring approximation algorithm.

In addition, if  $\text{period}(s) \leq \text{period}(\alpha)$ , then  $\text{ov}(s, \alpha[k]) < \frac{2}{3}(\text{period}(s) + \text{period}(\alpha))$ .

Our proof is *constructive*; it requires two computations of *critical factorizations* done in time that is linear in  $\text{period}(\alpha)$ . From now on, let  $\vec{\alpha}$  denote a rotation of  $\alpha$  satisfying Lemma 2. The bound in the last lemma is roughly tight because for any rotation of the semi-infinite string  $(0^n 10^{n+1} 1)^\infty$ , there exists a string with period at most  $n + 2$  which overlaps with  $(0^n 10^{n+1} 1)^\infty$  by at least  $2n + 2$ .

### 3. Approximation algorithms

Our algorithms are only slightly different from the ones in [1, 2, 3, 4, 5, 7, 10]. The main steps are shown in Figure 1. We show first that this generic algorithm has approximation ratio 3. Noting that

$$\langle s_{i_j}, \dots, s_{i_r}, s_{i_1}, \dots, s_{i_{j-1}}, s_{i_j} \rangle = \text{factor}(\langle s_{i_j}, \dots, s_{i_r}, s_{i_1}, \dots, s_{i_{j-1}} \rangle) s_{i_j},$$

it is straightforward [4, 10] that:  $\text{opt}(T) \leq \text{opt}(S) + \text{Cyc}(G_S) \leq 2 \text{opt}(S)$ ; hence, we have  $\text{Cyc}(G_T) \leq \text{opt}(T) \leq 2 \text{opt}(S)$ . We make use of the following upper bound on the possible overlap between two inequivalent strings  $s$  and  $t$ :  $\text{ov}(s, t) \leq \text{period}(s) + \text{period}(t)$ , and show that it applies to the strings in  $T$ . Then the total overlap represented by the edges broken in Step 6,  $OV$ , is at most the sum of the periods of the strings in  $T$ . By Corollary 1,  $OV \leq \sum_{c \in C} w(c) = \text{Cyc}(G_S)$ .

Putting everything together, we can bound the length of the superstring  $\tilde{s}$  as

$$|\tilde{s}| = \text{Cyc}(G_T) + OV \leq \text{Cyc}(G_T) + \text{Cyc}(G_S) \leq 2 \text{opt}(S) + \text{opt}(S) \leq 3 \text{opt}(S).$$

*The  $2\frac{2}{3}$ -approximation algorithm.* Many researchers have tried to improve the performance of the generic algorithm by polishing Steps 5 - 7. Nevertheless, Armen and Stein [1, 2] identified strings that are not much longer than their factors as the bottleneck and tried to avoid them in Step 3. A key difference between our algorithm and all the previous ones actually is Step 3. The previous algorithms all choose one of the strings contained in the cycle  $c$ , whereas here we look for a superstring of the strings in  $c$  that is not *too long*, to reduce  $OV$ . More precisely, we rely on:

LEMMA 3. For any cycle  $c = s_{i_1}, \dots, s_{i_r}, s_{i_1} \in C$ , there exists a string  $t_c$  such that for some  $j$ ,

- (1)  $t_c$  contains the string  $\langle s_{i_{j+1}}, \dots, s_{i_r}, s_{i_1}, \dots, s_{i_j} \rangle$ .
- (2)  $t_c$  is contained in the string  $\langle s_{i_j}, \dots, s_{i_r}, s_{i_1}, \dots, s_{i_{j-1}}, s_{i_j} \rangle$ .
- (3)  $(t_c)_\infty = \langle \overrightarrow{s_{i_1}, \dots, s_{i_r}} \rangle_\infty$ .

The string  $t_c$  can be found in linear time. We polish the generic algorithm by choosing  $t_c$  in Step 3 and changing step 6 into: For each cycle of  $CC$ , break the cycle by deleting an edge that goes from a string to a string of *equal or larger period*, to obtain a superstring of the elements in the cycle.

Note that we do not treat the small cycles of  $CC$  specially like the other algorithms do. Instead, we cut the cycles with a bit of care. Clearly, in every cycle there must be an edge that goes from a string to a string of equal or larger period. Applying Lemmas 2 and 1, we get

$$OV \leq \frac{2}{3} \sum_{c \in C} \text{period}(t_c) = \frac{2}{3} \sum_{c \in C} w(c) = \frac{2}{3} \text{Cyc}(G_S) \leq \frac{2}{3} \text{opt}(S).$$

Hence,  $|\bar{s}| = \text{Cyc}(G_T) + OV \leq 2\frac{2}{3} \text{opt}(S)$ .

*The  $2\frac{25}{42}$ -approximation algorithm.* Steps 5, 6 and 7 now become: *Construct a superstring of  $T$  using a good overlap approximation algorithm.* It was proven in [4] that the length  $\text{apx}(T)$  of the superstring of  $T$  produced by a  $\delta$  overlap approximation algorithm satisfies:  $\text{apx}(T) \leq \text{opt}(T) + (1 - \delta)\text{maxov}(T)$ . Our special choice of the cycle representatives  $t_c$  in Step 3 allows to improve on the standard bound used in all previous papers, e.g.  $\text{maxov}(T) \leq 2 \text{Cyc}(G_S)$ . By Lemma 2, we prove that:  $\text{maxov}(T) \leq \frac{3}{2} \text{Cyc}(G_S)$ . We use the  $\frac{38}{63}$  overlap approximation algorithm in [7], and get:  $\text{apx}(T) \leq \text{opt}(T) + (1 - \frac{38}{63})\text{maxov}(T) \leq 2 \text{opt}(S) + \frac{25}{63} \frac{3}{2} \text{Cyc}(G_S) \leq 2\frac{25}{42} \text{opt}(S)$ .

*Concluding remark.* We are still a long way from reaching the conjectured ratio 2 for approximating shortest superstrings.

### Bibliography

- [1] Armen (Chris) and Stein (Clifford). – Improved length bounds for the shortest superstring problem. In Akl (S. G.), Dehne (F.), Sack (J. R.), and Santoro (N.) (editors), *Algorithms and Data Structures. Proceedings. Lecture Notes in Computer Science*, pp. 494–505. – Berlin, Heidelberg, New York, 1995. 4th International Workshop, WADS '95, Kingston, Canada, 16–18 Aug. 1995.
- [2] Armen (Chris) and Stein (Clifford). – Short superstrings and the structure of overlapping strings. *Journal of Computational Biology*, 1995. – To appear.
- [3] Armen (Chris) and Stein (Clifford). – A  $2\frac{2}{3}$ -approximation algorithm for the shortest superstring problem. In *Combinatorial Pattern Matching. Proceedings. Lecture Notes in Computer Science*. – Berlin, Heidelberg, New York, 1996. 7th International Workshop.
- [4] Blum (A.), Jiang (T.), Li (M.), Tromp (J.), and Yanakakis (M.). – Linear approximation of shortest superstrings. *Journal of the ACM*, vol. 41, n° 4, 1994, pp. 630–647.
- [5] Czumaj (A.), Gąsieniec (L.), Piotrow (M.), and Rytter (W.). – Parallel and sequential approximation of shortest superstrings. In Schmidt (E. M.) and Skyum (S.) (editors), *Algorithm Theory - SWAT '94. Lecture Notes in Computer Science*, pp. 95–106. – Berlin, Heidelberg, New York, 1994. 4th Scandinavian Workshop on Algorithm Theory, Aarhus, Denmark, July 6–8, 1994.
- [6] Gallant (J.), Maier (D.), and Storer (J.). – On finding minimal length superstrings. *Journal of Computer System Sciences*, vol. 20, 1980, pp. 50–58.
- [7] Kosaraju (S. R.), Park (J.), and Stein (C.). – Long tours and short superstrings. In *Proceedings 35th IEEE Symposium on Foundations of Computer Science*. – 1994.
- [8] Papadimitriou (Christos H.) and Steiglitz (Kenneth). – *Combinatorial optimization : algorithms and complexity*. – Prentice Hall, Englewood Cliffs, N. J., 1982.
- [9] Tarhio (J.) and Ukkonen (E.). – A greedy approximation algorithm for constructing shortest common superstrings. *Theoretical Computer Science*, vol. 57, 1988, pp. 131–145.
- [10] Teng (S. H.) and Yao (F.). – Approximating shortest superstrings. In *Proceedings 34th IEEE Symposium on Foundations of Computer Science*, pp. 158–165. – 1993.
- [11] Turner (J.). – Approximation algorithms for the shortest common superstring problem. *Information and Computation*, vol. 83, 1989, pp. 1–20.