All Instantiations of the Greedy Algorithm for the Shortest Common Superstring Problem are Equivalent

Maksim S. Nikolaev(✉)

Steklov Institute of Mathematics at St. Petersburg, Russian Academy of Sciences, Saint Petersburg, Russia

Abstract. In the Shortest Common Superstring problem (SCS), one needs to find the shortest superstring for a set of strings. While SCS is NP-hard and MAX-SNP-hard, the Greedy Algorithm "choose two strings with the largest overlap; merge them; repeat" achieves a constant factor approximation that is known to be at most 3.5 and conjectured to be equal to 2. The Greedy Algorithm is not deterministic, so its instantiations with different tie-breaking rules may have different approximation factors. In this paper, we show that it is not the case: all factors are equal. To prove this, we show how to transform a set of strings so that all overlaps are different whereas their ratios stay roughly the same.

Keywords: Superstring · Shortest common superstring · Approximation · Greedy algorithms · Greedy conjecture

1 Introduction

In the Shortest Common Superstring problem (SCS), one is given a set of strings and needs to find the shortest string that contains each of them as a substring. Applications of this problem include genome assembly [12,19] and data compression [3,4,15]. We refer the reader to the survey [5] for an overview of SCS as well as its applications and algorithms.

SCS is known to be NP-hard [4] and even MAX-SNP-hard [1], but it admits constant-factor approximation in polynomial time. The best known approximation ratios are $2\frac{11}{23}$ due to Mucha [11] (see [7, Section 2.1] for an overview of the previous approximation algorithms and inapproximability results). While these approximation algorithms use many sophisticated techniques, the 30 years old *Greedy Conjecture* [1,15–17] claims that the trivial *Greedy Algorithm* (GA) "choose two strings with the largest overlap; merge them; repeat" is a factor 2 approximation (in fact, this is the best possible approximation factor: consider a dataset $\mathcal{S} = \{c(ab)^n, (ab)^n c, (ba)^n\}$). Ukkonen [18] shows that for a fixed alphabet, GA can be implemented in linear time.

The research is supported by Russian Science Foundation (18-71-10042).

T. Lecroq and H. Touzet (Eds.): SPIRE 2021, LNCS 12944, pp. 61–67, 2021.
https://doi.org/10.1007/978-3-030-86692-1_6

Blum et al. [1] prove that GA returns a 4-approximation of SCS, and Kaplan and Shafrir [8] improve this bound to 3.5. A slight modification of GA gives a 3-approximation of SCS [1], and other greedy algorithms are studied from theoretical [1,13] and practical perspectives [2,14].

It is known that the Greedy Conjecture holds for the case when all input strings have length at most 4 [9]. Also, the Greedy Conjecture holds if GA happens to merge strings in a particular order [10,20]. GA gives a 2-approximation of a different metric called compression [16]. The compression is defined as the sum of the lengths of all input strings minus the length of a superstring (hence, it is the number of symbols saved with respect to a naive superstring resulting from concatenating the input strings).

GA is not deterministic as we do not specify how to break ties in case when there are many pairs of strings with maximum overlap. For this reason, different instantiations of GA (that is, GA with a tie-breaking rule) may produce different superstrings for the same input and hence they may have different approximation factors. In fact, if $\mathcal{S}$ contains only strings of length 2 or less or if $\mathcal{S}$ is a set of k-substrings of an unknown string, then there are instantiations of GA [6], that find *the exact* solution, whereas in general GA fails to do so.

The original Greedy Conjecture states that *any* instantiation of GA is a factor 2 approximation. As this is still widely open, it is natural to try to prove the conjecture at least for *some* instantiations. This could potentially be easier not just because this is a weaker statement, but also because a particular instantiation of GA may decide how to break ties by asking an omniscient oracle. In this paper, we show that this weak form of Greedy Conjecture is in fact equivalent to the original one. More precisely, we show, that if *some* instantiation of GA is a factor λ approximation, then *all* instantiations are factor λ approximation.

To prove this, we introduce the so-called *Perturbing Procedure*, that, for a given dataset $\mathcal{S} = \{s_1, \ldots, s_n\}$, a parameter $m \gg n$, and a sequence of greedy non-trivial merges (merges of strings with a non-empty overlap), constructs a new dataset $\mathcal{S}' = \{s'_1, \ldots, s'_n\}$, such that, for all $i \neq j$, s'_i is roughly m times longer than s_i, the overlap of s'_i and s'_j is roughly m times longer than the overlap of s_i and s_j, and the mentioned greedy sequence of non-trivial merges for $\mathcal{S}$ is *the only* such sequence for $\mathcal{S}'$.

2 Preliminaries

Let $|s|$ be the length of a string s and $\mathrm{ov}(s,t)$ be the *overlap* of strings s and t, that is, the longest string y, such that $s = xy$ and $t = yz$. In this notation, a string xyz is *a merge* of strings s and t. By ε we denote the empty string. By $\mathrm{OPT}(\mathcal{S})$ we denote an optimal superstring for the dataset $\mathcal{S}$.

Without loss of generality we may assume that the set of input strings $\mathcal{S}$ contains no string that is a substring of another. This assumption implies that in any superstring all strings occur in some order: if one string begins before another, then it also ends before. Hence, we can consider only superstrings that

can be obtained from some permutation $(s_{\sigma(1)}, \ldots, s_{\sigma(n)})$ of $\mathcal{S}$ after merging adjacent strings. The length of such superstring $s(\sigma)$ is simply

$$|s(\sigma)| = \sum_{i=1}^{n} |s_i| - \sum_{i=1}^{n-1} |\operatorname{ov}\left(s_{\sigma(i)}, s_{\sigma(i+1)}\right)|. \tag{1}$$

Let A be an instantiation of GA (we denote this by $A \in \mathrm{GA}$). By σ_A we denote the permutation corresponding to a superstring $A(\mathcal{S})$ constructed by A, and by $(l_A(1), r_A(1)), \ldots, (l_A(n-1), r_A(n-1))$, we denote the order of merges: strings $s_{l_A(i)}$ and $s_{r_A(i)}$ are merged at step i. By the definition of GA we have

$$|\operatorname{ov}\left(s_{l_A(i)}, s_{r_A(i)}\right)| \geq |\operatorname{ov}\left(s_{l_A(j)}, s_{r_A(j)}\right)|, \quad \forall\, i < j < n,$$

and if, for some i, $|\operatorname{ov}(s_{l_A(i)}, s_{r_A(i)})| = 0$, then the same holds for any $i' > i$. We denote the first such i by T_A and this is the first trivial merge (that is, one with the empty overlap), after which all the merges are trivial. Note that just before step T_A, all the remaining strings have empty overlaps, so the resulting superstring is just a concatenation of them in some order and this order does not affect the length of the result. If there were no trivial merges, we set $T_A = n$.

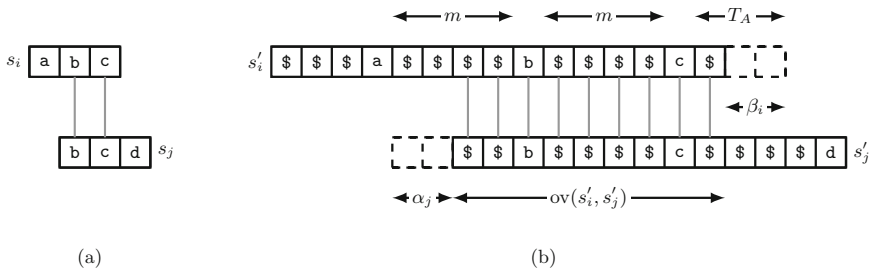

Fig. 1. (a) strings s_i and s_j from $\mathcal{S}$. (b) the resulting strings s'_i and s'_j after perturbing; here, $m = 4$, $T_A = 3$, $\alpha_i = 1$, $\beta_i = 2$, $\alpha_j = 2$ and $\beta_j = T_A$; since $\alpha_j = \beta_i = 2$, we may conclude that s_i and s_j were merged by A at step 2.

3 Perturbing Procedure

Here, we describe the mentioned procedure that eliminates ties. Consider a dataset $\mathcal{S}$, an instantiation $A \in \mathrm{GA}$ and *a sentinel* \$—a symbol that does not occur in $\mathcal{S}$, and a parameter m whose value will be determined later. For every string $s_i = c_1 c_2 \ldots c_{n_i} \in \mathcal{S}$ define a string

$$s'_i = \$^{m-\alpha_i} c_1 \$^m c_2 \$^m c_3 \$^m \ldots \$^m c_{n_i} \$^{T_A - \beta_i}, \tag{2}$$

where

1. α_i is the number of step such that $r_A(\alpha_i) = i$, if such step exists and is less than T_A, and $\alpha_i = T_A$ otherwise; note that if $\alpha_i < T_A$ then s_i is *the right* part of a non-trivial merge at step α_i;
2. β_i is the number of step such that $l_A(\beta_i) = i$, if such step exists and is less than T_A, and $\beta_i = T_A$ otherwise; note that if $\beta_i < T_A$ then s_i is *the left* part of a non-trivial merge at step β_i.

Basically, we insert the string $\$^m$ before every character of s_i and then remove some \$'s from the beginning of the string and add some \$'s to its end (see Fig. 1). The purpose of this removal and addition is to perturb slightly overlaps of equal length, so there are no longer any ties in non-trivial merges.

We denote the resulting set of perturbed strings $\{s'_1, \dots, s'_n\}$ by $\mathcal{S}'$, and all entities related to this dataset we denote by adding a prime (for example, σ'_A). Let us derive some properties of $\mathcal{S}'$.

Lemma 1. *For all $i \neq j$, $k \neq l$, $m > 2n$*

1. *if $|\operatorname{ov}(s_i, s_j)| = d > 0$, then $|\operatorname{ov}(s'_i, s'_j)| = (m+1)d - \alpha_j + T_A - \beta_i$;*
2. *if $|\operatorname{ov}(s_i, s_j)| = 0$, then $|\operatorname{ov}(s'_i, s'_j)| = \min\{T_A - \beta_i, m - \alpha_j\}$;*
3. *perturbing procedure preserves order on overlaps of different lengths, that is, if $|\operatorname{ov}(s_i, s_j)| > |\operatorname{ov}(s_k, s_l)|$, then $|\operatorname{ov}(s'_i, s'_j)| > |\operatorname{ov}(s'_k, s'_l)|$.*

Proof. Let $\operatorname{ov}(s_i, s_j)$ be $c_1 c_2 \dots c_d$. Consider the string

$$u = \$^{m-\alpha_j} c_1 \$^m \dots \$^m c_d \$^{T_A - \beta_i}.$$

Clearly, u is the overlap of s'_i and s'_j and $|u| = (m+1)d - \alpha_j + T_A - \beta_i$. Also, if $|\operatorname{ov}(s_i, s_j)| = 0$ then $\operatorname{ov}(s'_i, s'_j) = \$^{\min\{T_A - \beta_i, m - \alpha_j\}}$.

To prove the last statement, note that $\alpha_j + \beta_i \leq 2T_A < m$ and

$$|\operatorname{ov}\left(s'_i, s'_j\right)| > (m+1)|\operatorname{ov}\left(s_k, s_l\right)| + T_A \geq |\operatorname{ov}\left(s'_k, s'_l\right)|.$$

Lemma 2. *Let $B \in \mathrm{GA}$. Then $T_A = T'_A = T'_B$ and the first $T_A - 1$ merges are the same for both instantiations.*

Proof. We prove by induction that $l_A(t) = l'_A(t) = l'_B(t)$ and $r_A(t) = r'_A(t) = r'_B(t)$ for all $t < T_A$.

Case $t = 1$. As A is greedy, then $k_1 := |\operatorname{ov}(s_{l_A(1)}, s_{r_A(1)})| \geq |\operatorname{ov}(s_i, s_j)|$, for all $i \neq j$, $(i, j) \neq (l_A(1), r_A(1))$. Hence

$$\begin{aligned} |\operatorname{ov}\left(s'_i, s'_j\right)| &\leq (m+1)k_1 - \alpha_j + T_A - \beta_i \\ &< (m+1)k_1 - 1 + T_A - 1 = |\operatorname{ov}\left(s'_{l_A(1)}, s'_{r_A(1)}\right)|, \end{aligned}$$

and $l'_A(1) = l'_B(1) = l_A(1)$ as well as $r'_A(1) = r'_B(1) = r_A(1)$.

Suppose that the statement holds for all $t \leq t' < T_A - 1$. Note that at moment $t = t' + 1$ the sum $\alpha_j + \beta_i$ is strictly greater than $2t$ unless $(i, j) = (l_A(t), r_A(t))$. Similarly to the base case, we have

$$\begin{aligned} |\operatorname{ov}\left(s'_i, s'_j\right)| &\leq (m+1)k_t - \alpha_j + T_A - \beta_i \\ &< (m+1)k_t - t + T_A - t = |\operatorname{ov}\left(s'_{l_A(t)}, s'_{r_A(t)}\right)|, \end{aligned}$$

where $k_t = |\operatorname{ov}(s_{l_A(t)}, s_{r_A(t)})|$, and the induction step is proven.

Now note that starting from step T_A all the remaining strings in $\mathcal{S}$ have empty overlaps and hence so do the remaining strings in $\mathcal{S}'$, as for all of them $\beta_i = T_A$ and the minimum in paragraph 2 of Lemma 1 is equal to zero. Thus, $T_A = T'_A = T'_B$ and the lemma is proven.

Corollary 1. *As all non-trivial merges coincide,* $|A(\mathcal{S}')| = |B(\mathcal{S}')|$.

4 Equivalence of Instantiations

Theorem 1. *If some instantiation A of GA achieves a λ-approximation, then so does any other instantiation.*

Proof. Assume the opposite and consider $B \in \text{GA}$ as well as a dataset $\mathcal{S}$ such that $|B(\mathcal{S})| > \lambda|\text{OPT}(\mathcal{S})|$. Let $\mathcal{S}' = \mathcal{S}'(B, m)$ be the corresponding perturbed dataset, where $m > 2n$ will be specified later.

Note that $|s'_i|/m \to |s_i|$ and $|\operatorname{ov}(s'_i, s'_j)|/m \to |\operatorname{ov}(s_i, s_j)|$ as m approaches infinity, thanks to Lemma 1.1–2. Then $|\text{OPT}(\mathcal{S}')|/m \to |\text{OPT}(\mathcal{S})|$, since

$$\begin{aligned}\frac{1}{m}|\text{OPT}(\mathcal{S}')| &= \frac{1}{m}\min_{\sigma}\left\{\sum_{i=1}^{n}|s'_i| - \sum_{i=1}^{n-1}|\operatorname{ov}\left(s'_{\sigma(i)}, s'_{\sigma(i+1)}\right)|\right\} \\ &\to \min_{\sigma}\left\{\sum_{i=1}^{n}|s_i| - \sum_{i=1}^{n-1}|\operatorname{ov}\left(s_{\sigma(i)}, s_{\sigma(i+1)}\right)|\right\} = |\text{OPT}(\mathcal{S})|,\end{aligned}$$

$|B(\mathcal{S}')|/m \to |B(\mathcal{S})|$ and hence $|A(\mathcal{S}')|/m \to |B(\mathcal{S})|$, by Corollary 1.

As $|B(\mathcal{S})| - \lambda|\text{OPT}(\mathcal{S})| > 0$, we can choose m so that $|B(\mathcal{S}')| - \lambda|\text{OPT}(\mathcal{S}')|$ as well as $|A(\mathcal{S}')| - \lambda|\text{OPT}(\mathcal{S}')|$ are positive. Hence A is not a factor λ approximation.

Corollary 2. *To prove (or disprove) the Greedy Conjecture, it is sufficient to consider datasets satisfying some of the following three properties:*

1. *there are no ties between non-empty overlaps, that is, datasets where all the instantiations of the greedy algorithm work the same;*
2. *there are no empty overlaps:* $\operatorname{ov}(s_i, s_j) \neq \varepsilon$, $\forall\, i \neq j$;
3. *all non-empty overlaps are (pairwise) different:* $|\operatorname{ov}(s_i, s_j)| \neq |\operatorname{ov}(s_k, s_l)|$, *for all* $i \neq j$, $k \neq l$, $(i, j) \neq (k, l)$.

Proof. 1. Follows directly from the proof of Theorem 1, as we always can use the dataset $\mathcal{S}'$ instead of $\mathcal{S}$.

2. Append \$ to each string of $\mathcal{S}'$. Then, every two strings have non-empty overlap that at least contains \$, and in general $T_A = T'_A = T'_B$ from Lemma 2 does not hold (T'_A and T'_B are always n). However, the first T_A merges are still the same and after them all the remaining strings have overlaps of length 1 and then the lengths of the final solutions are the same as well.

3. Append $\$^{n(T_A-\beta_i)}$ to each string of $\mathcal{S}'$ instead of $\$^{T_A-\beta_i}$. Then

$$|\operatorname{ov}(s'_i, s'_j)| = (m+1)|\operatorname{ov}(s_i, s_j)| - \alpha_j + nT_A - n\beta_i,$$

provided m is large enough, and $\alpha_j + n\beta_i \neq \alpha_k + n\beta_l$ if $(i,j) \neq (k,l)$. Repeating the proofs of Lemmas 1 and 2 with this version of $\mathcal{S}'$, we obtain this statement of the corollary.

To combine several of this properties (for example second and third), it is sufficient to sequentially apply the corresponding transformations on the original dataset: at first we get a dataset $\mathcal{S}'$ from $\mathcal{S}$ as in paragraph 2, then we treat $\mathcal{S}'$ (already without empty overlaps) as original and transform it to a dataset $\mathcal{S}''$ according to paragraph 3 using a different sentinel instead of \$.

5 Conclusion

In this paper we revealed the equivalence of greedy algorithms for the shortest common superstring problem. This means, in particular, that proving or disproving the Greedy Conjecture is difficult not due to the non-deterministic nature of the Greedy Algorithm, but due to the complexity of the overlaps structure.

Acknowledgments. Many thanks to Alexander Kulikov for valuable discussions and proofreading the text, and the anonymous reviewers for their useful comments.

References

1. Blum, A., Jiang, T., Li, M., Tromp, J., Yannakakis, M.: Linear approximation of shortest superstrings. In: STOC 1991, pp. 328–336. ACM (1991). https://doi.org/10.1145/179812.179818
2. Cazaux, B., Juhel, S., Rivals, E.: Practical lower and upper bounds for the shortest linear superstring. In: SEA 2018, vol. 103, pp. 18:1–18:14. LIPIcs (2018). https://doi.org/10.4230/LIPIcs.SEA.2018.18
3. Gallant, J.K.: String compression algorithms. Ph.D. thesis, Princeton (1982)
4. Gallant, J., Maier, D., Storer, J.A.: On finding minimal length superstrings. J. Comput. Syst. Sci. **20**(1), 50–58 (1980). https://doi.org/10.1016/0022-0000(80)90004-5
5. Gevezes, T.P., Pitsoulis, L.S.: The shortest superstring problem. In: Rassias, T.M., Floudas, C.A., Butenko, S. (eds.) Optimization in Science and Engineering, pp. 189–227. Springer, New York (2014). https://doi.org/10.1007/978-1-4939-0808-0_10
6. Golovnev, A., Kulikov, A.S., Logunov, A., Mihajlin, I., Nikolaev, M.: Collapsing superstring conjecture. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2019). https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2019.26
7. Golovnev, A., Kulikov, A.S., Mihajlin, I.: Approximating shortest superstring problem using de bruijn graphs. In: Fischer, J., Sanders, P. (eds.) CPM 2013. LNCS, vol. 7922, pp. 120–129. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38905-4_13

8. Kaplan, H., Shafrir, N.: The greedy algorithm for shortest superstrings. Inf. Process. Lett. **93**(1), 13–17 (2005). https://doi.org/10.1016/j.ipl.2004.09.012
9. Kulikov, A.S., Savinov, S., Sluzhaev, E.: Greedy conjecture for strings of length 4. In: Cicalese, F., Porat, E., Vaccaro, U. (eds.) CPM 2015. LNCS, vol. 9133, pp. 307–315. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19929-0_26
10. Laube, U., Weinard, M.: Conditional inequalities and the shortest common superstring problem. Int. J. Found. Comput. Sci. **16**(06), 1219–1230 (2005). https://doi.org/10.1142/S0129054105003777
11. Mucha, M.: Lyndon words and short superstrings. In: SODA 2013, pp. 958–972. SIAM (2013). https://doi.org/10.1137/1.9781611973105.69
12. Pevzner, P.A., Tang, H., Waterman, M.S.: An Eulerian path approach to DNA fragment assembly. Proc. Natl. Acad. Sci. U.S.A. **98**(17), 9748–9753 (2001). https://doi.org/10.1073/pnas.171285098
13. Rivals, E., Cazaux, B.: Superstrings with multiplicities. In: CPM 2018, vol. 105, pp. 21:1–21:16 (2018). https://doi.org/10.4230/LIPIcs.CPM.2018.21
14. Romero, H.J., Brizuela, C.A., Tchernykh, A.: An experimental comparison of two approximation algorithms for the common superstring problem. In: ENC 2004, pp. 27–34. IEEE (2004). https://doi.org/10.1109/ENC.2004.1342585
15. Storer, J.A.: Data compression: methods and theory. Computer Science Press Inc., (1987)
16. Tarhio, J., Ukkonen, E.: A greedy approximation algorithm for constructing shortest common superstrings. Theor. Comput. Sci. **57**(1), 131–145 (1988). https://doi.org/10.1016/0304-3975(88)90167-3
17. Turner, J.S.: Approximation algorithms for the shortest common superstring problem. Inf. Comput. **83**(1), 1–20 (1989). https://doi.org/10.1016/0890-5401(89)90044-8
18. Ukkonen, E.: A linear-time algorithm for finding approximate shortest common superstrings. Algorithmica **5**(1–4), 313–323 (1990). https://doi.org/10.1007/BF01840391
19. Waterman, M.S.: Introduction to Computational Biology: Maps, Sequences and Genomes. CRC Press, Boca Raton (1995). https://doi.org/10.1201/9780203750131
20. Weinard, M., Schnitger, G.: On the greedy superstring conjecture. SIAM J. Discret. Math. **20**(2), 502–522 (2006). https://doi.org/10.1137/040619144