Generalized Greedy Algorithm for Shortest Superstring

Zhengjun Cao[1,2], Lihua Liu[3], and Olivier Markowitch[2]

[1] Department of Mathematics, Shanghai University, Shanghai, China
caoamss@gmail.com
[2] Department of Computer Sciences, Université Libre de Bruxelles, Belgium
[3] Department of Mathematics, Shanghai Maritime University, China

Abstract. In the primitive greedy algorithm for shortest superstring, if a pair of strings with maximum overlap picked out, they are subsequently merged. In this paper, we introduce the concept of optimal set and generalize the primitive greedy algorithm. The generalized algorithm can be reduced to the primitive greedy algorithm if the relative optimal set is empty. Consequently, the new algorithm achieves a better bound at the expense of cost. But the cost is acceptable in practice.

Keywords: greedy algorithm, shortest superstring, optimal set.

1 Introduction

It's well known that the human DNA can be viewed as a very long string over a four-letter alphabet. Lots of scientists are attempting to decipher this string. Since it is very long, several overlapping short segments of this string are first deciphered. But the locations of these segments on the original DNA are not known. It is hypothesized that the shortest string which contains these segments as substrings is a good approximation to the original DNA string [6,7,9]. This leads to the shortest superstring problem:

Given a finite alphabet $\sum$*, and a set of* n *strings,* $S = \{s_1, \cdots, s_n\} \subseteq \sum^*$*, find a shortest string* s *that contains each* s_i *as a substring.*

The shortest superstring problem is NP-hard [8]. There are many algorithms [1-11] for the problem. Perhaps the following three algorithms are more attractive.

The primitive greedy algorithm. The first algorithm that comes to mind for finding a short superstring is the following primitive greedy algorithm (GREEDY for short). Define the *overlap* of two string $s, t \in \sum^*$ as the maximum length of a suffix of s that is also a prefix of t. The algorithm maintains a set of strings T; initially $T = S$. At each step, the algorithm selects from T two strings that have maximum overlap and replaces them with the string obtained by overlapping them as much as possible. After $n - 1$ steps, T will contain a single string. This algorithm is conjectured to have an approximation factor of 2.

The greedy set covering algorithm. To see that the approximation factor of Greedy algorithm is no better than 2, consider an input consisting of 3 strings:

J. Zhou (Ed.): Complex 2009, Part II, LNICST 5, pp. 1526–1531, 2009.

ab^k, b^kc, and b^{k+1}. If the first two strings are selected in the first iteration, the Greedy algorithm produces the string ab^kcb^{k+1}. This is almost twice as long as the shortest superstring, $ab^{k+1}c$. We will obtain a $2H_n$ factor approximation algorithm, using the greedy set cover algorithm [11], where $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$.

The greedy cycle covering algorithm. Blum et al. [2] have proposed an improvement of Greedy for DNA shortest superstrings. They called it T-Greedy. It produces a superstring of length at most $3 \cdot OPT(s)$.

All greedy algorithms mentioned above do repeatedly merge a pair of strings with maximum overlap until only one string left. Once a pair of strings with maximum overlap picked out in them, they are directly merged even though there are some special strings can be merged between them such that the resulting string is of more shorter length.

In this paper, we introduce the concept of optimal set, and generalize GREEDY by merging some optimal strings between the early pair of strings with maximum overlap. If the relative optimal set is empty, the generalized greedy algorithm can be reduced to GREEDY. This grantees that the new algorithm achieves a better bound than GREEDY. The cost of the new algorithm is a little greater than GREEDY. But it is still less than the greedy set covering algorithm or the greedy cycle covering algorithm, because the new algorithm need not compute a distance graph and find a minimum length cycle cover.

This paper is organized as follows. In Section 2, we describe the basic idea and put forward the concept of optimal set. The description of the generalized greedy algorithm is presented in Section 3. In Section 4, we provide two examples to explain the new algorithm in detail. The conclusion is given in Section 5.

2 Basic Idea and Optimal Set

Let $\mathcal{M} = \{m_1, \cdots, m_n\}$ be a set of strings over some alphabet Σ, $|\mathcal{M}|$ be the cardinal number of $\mathcal{M}$, $|m|$ be the length of string m. Given two strings α and β, assume that γ is the longest string such that $\alpha = x\gamma$ and $\beta = \gamma z$. γ is called the maximum overlap between α and β. It is denoted by $ov(\alpha, \beta)$. Let $\alpha \circ \beta$ denote the new string by merging the maximum overlap between α and β.

The basic idea behind the generalized greedy algorithm (G-GREEDY for short) is to construct a set $O_{(\alpha,\beta)}$ related to the pair (α, β) such that

$$|\alpha \circ \gamma \circ \beta| \leq \min\{|\alpha \circ \beta \circ \gamma|, |\gamma \circ \alpha \circ \beta|\}$$

where $\gamma \in O_{(\alpha,\beta)}$. After the construction of the set, we choose an optimal string from the set, denoted by λ, and merge λ with α or β, instead of forming the new string $\alpha \circ \lambda \circ \beta$ directly. This ensures that GREEDY can be reduced to GREEDY in the trivial case, namely, $O_{(\alpha,\beta)} = \emptyset$.

Definition 2.1. Given a set $\mathcal{M}$ of strings and a pair of strings $\alpha, \beta \in \mathcal{M}$, we call

$$O_{(\alpha,\beta)} \stackrel{\text{def}}{=} \{\, x \,:\, |\alpha \circ x \circ \beta| \leq \min\{|x \circ \alpha \circ \beta|, |\alpha \circ \beta \circ x|\},\ x \in M \backslash \{\alpha, \beta\}\}$$

the *optimal set* w.r.t. (α, β) in $\mathcal{M}$. $x \in O_{(\alpha,\beta)}$ is called *optimal string* relative to (α, β).

3 Generalized Greedy Algorithm

3.1 Description

Now we introduce a new algorithm for shortest superstring based on the concept of optimal set.

- Input: $\mathcal{M}$.
- Step-1. [Deletion] Delete the strings of $\mathcal{M}$ such that it is substring-free.
- Step-2. [Global maximum overlap principle] Pick out all pairs with maximum overlap, denoted by
$$\Lambda = \{(\alpha_1, \beta_1), \cdots, (\alpha_i, \beta_i)\}$$
- Step-3. [Local minimum string principle] Choose a pair $(\alpha, \beta) \in \Lambda$ such that
$$|\alpha \circ \beta| = \min\{|\alpha_1 \circ \beta_1|, \cdots, |\alpha_i \circ \beta_i|\}$$
- Step-4. [Computation of optimal set] Compute the optimal set
$$O_{(\alpha,\beta)} = \{\gamma_1, \cdots, \gamma_j\}$$
- Step-5. [Local maximum overlap principle] If $O_{(\alpha,\beta)} \neq \emptyset$, let
$$\Psi = \{(\alpha, \gamma_k), (\gamma_k, \beta), \quad k = 1, \cdots, j\}$$
Choose the pair $(\mu, \nu) \in \Psi$ such that
$$|ov(\mu, \nu)| = \max\{|ov(\alpha, \gamma_k)|, |ov(\gamma_k, \beta)|, \quad k = 1, \cdots, j\}$$
Add $\mu \circ \nu$ to $\mathcal{M}$ and update it by Deletion. If $|\mathcal{M}| > 1$, goto step-2. Otherwise output the single string.
- Step-6. If $O_{(\alpha,\beta)} = \emptyset$, then add $\alpha \circ \beta$ to $\mathcal{M}$. Update $\mathcal{M}$ by Deletion. If $|\mathcal{M}| > 1$, goto step-2. Otherwise output the single string.

3.2 Complexity

Compared with GREEDY, the extra cost of G-GREEDY is mainly dominated by the computation of optimal set. Concretely, suppose that the original set (substring-free) is of n strings, to pick out a pair of strings with maximum overlap, both GREEDY and G-GREEDY should generate $n(n-1)$ new strings and compute their lengths. In step-4, G-GREEDY has to generate extra $3(n-2)$ strings and compute their lengths. Therefore, the cost of G-GREEDY is linear with respect to n. It's acceptable in practice. Clearly, the cost of G-GREEDY is still less than the greedy cycle cover algorithm, because the latter has to build a distance graph and search for the minimum length cycle cover of the corresponding graph.

4 Examples for G-GREEDY

We now provide some examples to explain G-Greedy in detail.

Example 1. $\mathcal{M} = \{c^k ab^k, ab^k ab^k a, b^k dab^{k-1}, ab^k c^{k+1}\}$

Phase 4$\Longrightarrow$ 3.

[Global maximum overlap principle] Pick out all pairs with maximum overlap, we have

$$\{(c^k ab^k, ab^k ab^k a), (c^k ab^k, ab^k c^{k+1})\}$$

[Local minimum string principle] Since

$$|c^k ab^k \circ ab^k ab^k a| = |c^k ab^k ab^k a| = 3k+3,\ |c^k ab^k \circ ab^k c^{k+1}| = |c^k ab^k c^{k+1}| = 3k+2$$

we pick the pair $(c^k ab^k, ab^k c^{k+1})$.

[Computation of optimal set] Since

$$\begin{aligned}
&|c^k ab^k \circ \underline{ab^k ab^k a} \circ ab^k c^{k+1}| = 5k+4\\
&|\underline{ab^k ab^k a} \circ c^k ab^k \circ ab^k c^{k+1}| = 5k+5\\
&|c^k ab^k \circ ab^k c^{k+1} \circ \underline{ab^k ab^k a}| = 5k+5;\\
&|c^k ab^k \circ \underline{b^k dab^{k-1}} \circ ab^k c^{k+1}| = 4k+4\\
&|\underline{b^k dab^{k-1}} \circ c^k ab^k \circ ab^k c^{k+1}| = 5k+3\\
&|c^k ab^k \circ ab^k c^{k+1} \circ \underline{b^k dab^{k-1}}| = 5k+3
\end{aligned}$$

we have

$$O_{(c^k ab^k, ab^k c^{k+1})} = \{ab^k ab^k a, b^k dab^{k-1}\}$$

[Local maximum overlap principle] Since

$$\begin{aligned}
&|ov(c^k ab^k, ab^k ab^k a)| = |ab^k| = k+1,\ |ov(ab^k ab^k a, ab^k c^{k+1})| = |a| = 1\\
&|ov(c^k ab^k, b^k dab^{k-1})| = |b^k| = k,\ |ov(b^k dab^{k-1}, ab^k c^{k+1})| = |ab^{k-1}| = k
\end{aligned}$$

we choose the pair $(c^k ab^k, ab^k ab^k a)$. Merge the pair and add the string $c^k ab^k ab^k a$ to $\mathcal{M}$. Hence, we have

$$\{c^k ab^k ab^k a, b^k dab^{k-1}, ab^k c^{k+1}\}$$

Phase 3$\Longrightarrow$2. Pick out all pairs with maximum overlap, we have

$$\{(ab^k c^{k+1}, c^k ab^k ab^k a), (b^k dab^{k-1}, ab^k c^{k+1})\}$$

Since

$$|ab^k c^{k+1} \circ c^k ab^k ab^k a| = 4k+5,\ |b^k dab^{k-1} \circ ab^k c^{k+1}| = 3k+3$$

we choose the pair $(b^k dab^{k-1}, ab^k c^{k+1})$. Since

$$|b^k dab^{k-1} \circ \underline{c^k ab^k ab^k a} \circ ab^k c^{k+1}| = 7k+5,\ |b^k dab^{k-1} \circ ab^k c^{k+1} \circ \underline{c^k ab^k ab^k a}| = 5k+6$$

we have $O_{(b^k dab^{k-1}, ab^k c^{k+1})} = \emptyset$. Merge the pair $(b^k dab^{k-1}, ab^k c^{k+1})$ and add the string $b^k dab^k c^{k+1}$ to $\mathcal{M}$. Hence, we have

$$\{b^k dab^k c^{k+1}, c^k ab^k ab^k a\}$$

Thus, the resulting string is $b^k dab^k c^{k+1} ab^k ab^k a$, of length $5k + 6$.

Example 2. $\mathcal{M} = \{ate, half, lethal, alpha, alfalfa\}$.

Phase $5 \Longrightarrow 4$. Pick out all pairs with maximum overlap, we have

$$\{(half, alfalfa), (lethal, half)\}$$

Since $|half \circ alfalfa| = 8,\ |lethal \circ half| = 7$, we choose the pair $(lethal, half)$. Since

$|lethal \circ \underline{ate} \circ half| = 13, |\underline{ate} \circ lethal \circ half| = 10$

$|lethal \circ \underline{alfalfa} \circ half| = 15, |\underline{alfalfa} \circ lethal \circ half| = 14$

$|lethal \circ \underline{alpha} \circ half| = 11, |\underline{alpha} \circ lethal \circ half| = 12, |lethal \circ half \circ \underline{alpha}| = 12$

Thus $O_{(lethal, half)} = \{alpha\}$. Since

$$|ov(lethal, alpha)| = |al| = 2,\ |ov(alpha, half)| = |ha| = 2$$

we randomly choose a pair, $(lethal, alpha)$. Merge the pair and add the new string $lethalpha$ to $\mathcal{M}$. Hence, we have

$$\{ate, half, lethalpha, alfalfa\}$$

Phase $4 \Longrightarrow 3$. Pick out all pairs with maximum overlap, we have $\{(half, alfalfa)\}$. Since

$$|half \circ \underline{ate} \circ alfalfa| = 14, |half \circ alfalfa \circ \underline{ate}| = 10$$
$$|half \circ \underline{lethalpha} \circ alfalfa| = 19, |\underline{lethalpha} \circ half \circ alfalfa| = 15$$

Thus $O_{(half, alfalfa)} = \emptyset$. Merge the pair and add the new string $halfalfa$ to $\mathcal{M}$. Hence, we have

$$\{ate, halfalfa, lethalpha\}$$

Phase $3 \Longrightarrow 2$. Pick out all pairs with maximum overlap, we have

$$\{(halfalfa, ate), (lethalpha, ate)\}$$

Since $|halfalfa \circ ate| = 10, |lethalpha \circ ate| = 11$, we chose the pair $(halfalfa, ate)$ and compute the optimal set. Since

$$|halfalfa \circ \underline{lethalpha} \circ ate| > |\underline{lethalpha} \circ halfalfa \circ ate|$$

Thus $O_{(halfalfa, ate)} = \emptyset$. Merge the pair and add the new string $halfalfate$ to $\mathcal{M}$. Hence, we have

$$\{halfalfate, lethalpha\}$$

Thus, the resulting string is $lethalphalfalfate$, of length 17.

Remark 1. In the Example 2, if we choose the pair $(alpha, half)$ in the phase $5 \Longrightarrow 4$, we also obtain the same resulting string $lethalphalfalfate$. But $lethalfalfalphate$ is a shortest supersting, too. This shows G-GREEDY can not find some specific shortest superstrings.

5 Conclusion

Lots of experiments on G-GREEDY show that it always achieves a better bound than GREEDY. But it is hard to prove a theoretical bound for it at present. More interestingly, it remains open to disprove that G-GREEDY can reach a shortest superstring.

Acknowledgements

National Natural Science Foundation of China (Project 60873227), Innovation Program of Shanghai Municipal Education Commission.

References

1. Armen, C., Stein, C.: A 2 2/3-approximation algorithm for the shortest superstring problem. In: CPM, pp. 87–101 (1996)
2. Blum, A., Jiang, T., Li, M., Tromp, J., Yannakakis, M.: Linear approximation of shortest superstrings. Journal of ACM 4, 630–647 (1994)
3. Breslauer, D., Jiang, T., Jiang, Z.: Rotations of periodic strings and short superstrings. J. Algorithms 24(2), 340–353 (1997)
4. Czumaj, A., Gasieniec, L., Piotrow, M., Rytter, W.: Parallel and sequential approximations of shortest superstrings. In: Scandinavian Workshop on Algorithm Theory, pp. 95–106 (1994)
5. Kaplan, H., Shafrir, N.: The greedy algorithm for shortest superstrings. Inf. Process. Lett. 93(1), 13–17 (2005)
6. Lesk, A. (ed.): Computational Molecular Biology, sources and methods for sequence analysis. Oxford University Press, Oxford (1988)
7. Li, M.: Towards a DNA sequencing theory. In: 31st IEEE Symp. on Foundations of Computer Science, pp. 125–134 (1990)
8. Gallant, J., Maier, D., Storer, J.: On finding minimal length superstrings. Journal of Computer and System Sciences 20, 50–58 (1980)
9. Peltola, H., Soderlund, H., Tarhio, J., Ukkonen, E.: Algorithms for some string matching problems arising in molecular genetics. In: Proc. IFIP Congress, pp. 53–64 (1983)
10. Sweedyk, Z.: A 2 1/2-approximation algorithm for shortest superstring. SIAM J. Comput. 29(3), 954–986 (1999)
11. Vazirani, V.: Approximation algorithm, pp. 61–66. Springer, Heidelberg (2003)