Information Processing Letters 116 (2016) 245–251

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

On the greedy algorithm for the Shortest Common Superstring problem with reversals

Gabriele Fici [a,*], Tomasz Kociumaka [b], Jakub Radoszewski [b,c], Wojciech Rytter [b], Tomasz Waleń [b]

[a] *Dipartimento di Matematica e Informatica, Università di Palermo, Italy*
[b] *Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland*
[c] *Newton International Fellow at Department of Informatics, King's College London, UK*

ARTICLE INFO

Article history:
Received 3 July 2015
Received in revised form 26 November 2015
Accepted 26 November 2015
Available online 2 December 2015
Communicated by Jef Wijsen

Keywords:
Analysis of algorithms
Shortest Common Superstring
Reversal
Greedy algorithm

ABSTRACT

We study a variation of the classical Shortest Common Superstring (SCS) problem in which a shortest superstring of a finite set of strings S is sought containing as a factor every string of S *or* its reversal. We call this problem Shortest Common Superstring with Reversals (SCS-R). This problem has been introduced by Jiang et al. [9], who designed a greedy-like algorithm with length approximation ratio 4. In this paper, we show that a natural adaptation of the classical greedy algorithm for SCS has (optimal) *compression ratio* $\frac{1}{2}$, i.e., the sum of the overlaps in the output string is at least half the sum of the overlaps in an optimal solution. We also provide a linear-time implementation of our algorithm.
© 2015 Elsevier B.V. All rights reserved.

1. Introduction

The Shortest Common Superstring (SCS) problem is a classical combinatorial problem on strings with applications in many domains, e.g. DNA fragment assembly, data compression, etc. (see [6] for a recent survey). It consists, given a finite set of strings S over an alphabet Σ, in finding a shortest string containing as factors (substrings) all the strings in S. The decision version of the problem is known to be NP-complete [13,5,4], even under several restrictions on the structure of S (see again [6]). However, a particularly simple greedy algorithm introduced by Gallant in his Ph.D. thesis [5] is widely used in applications since it has very good performance in practice (see for instance [12] and references therein). It consists in repetitively replacing a pair of strings with maximum overlap with the string obtained by overlapping the two strings, until one string remains. The greedy algorithm can be implemented using Aho–Corasick automaton in $\mathcal{O}(n)$ randomized time (with hashing on the symbols of the alphabet) or $\mathcal{O}(n \min(\log m, \log|\Sigma|))$ deterministic time (see [17]), where n is the sum of the lengths of the strings in S and m its cardinality.

The approximation of the greedy algorithm is usually measured in two different ways: one consists in taking into account the *approximation ratio* (also known as the *length ratio*) k_g/k_{min}, where k_g is the length of the output string of greedy and k_{min} the length of a shortest superstring, the other consists in taking into account the *compression ratio* $(n-k_g)/(n-k_{min})$.

For the approximation ratio, Turner [16] proved that there is no constant $c < 2$ such that $k_g/k_{min} \le c$. The *greedy conjecture* states that this approximation ratio is in fact 2 [1]. The best bound currently known is 3.5 due to

* Corresponding author.
E-mail addresses: gabriele.fici@unipa.it (G. Fici), kociumaka@mimuw.edu.pl (T. Kociumaka), jrad@mimuw.edu.pl (J. Radoszewski), rytter@mimuw.edu.pl (W. Rytter), walen@mimuw.edu.pl (T. Waleń).

http://dx.doi.org/10.1016/j.ipl.2015.11.015
0020-0190/© 2015 Elsevier B.V. All rights reserved.

Kaplan and Shafrir [10]. Algorithms with better approximation ratio are known; the best one is due to Mucha, with an approximation ratio of $2\frac{11}{23}$ [14].

For the compression ratio, Tarhio and Ukkonen [15] proved that $(n-k_g)/(n-k_{min}) \geq \frac{1}{2}$ and this bound is tight, since it is achieved for the set $S=\{ab^h, b^h a, b^{h+1}\}$ when greedy makes the first choice merging the first two strings together.

Let us formally state the SCS problem:

> SHORTEST COMMON SUPERSTRING (*SCS*)
> **Input:** strings $S=\{s_1,\ldots,s_m\}$ of total length n.
> **Output:** a shortest string u that contains s_i for each $i=1,\ldots,m$ as a factor.

Several variations of SCS have been considered in literature. For example, shortest common superstring problem with reverse complements was considered in [11]. In this setting the alphabet is $\Sigma=\{\mathtt{a},\mathtt{t},\mathtt{g},\mathtt{c}\}$ and the complement of a string s is $\bar{s}^R$, where $\bar{}$ is defined by $\bar{\mathtt{a}}=\mathtt{t}$, $\bar{\mathtt{t}}=\mathtt{a}$, $\bar{\mathtt{g}}=\mathtt{c}$, $\bar{\mathtt{c}}=\mathtt{g}$, and t^R denotes the reversal of t, that is the string obtained reading t backwards. In particular, this problem was shown to be NP-complete.

Other variations of the SCS problem can be found in [8,3,7,2].

In this paper, we address the problem of searching for a string u of minimal length such that for every $s_i \in S$, u contains as a factor s_i *or* its reversal s_i^R.

> SHORTEST COMMON SUPERSTRING WITH REVERSALS (*SCS-R*)
> **Input:** strings $S=\{s_1,\ldots,s_m\}$ of total length n.
> **Output:** a shortest string u that contains for each $i=1,\ldots,m$ at least one of the strings s_i or s_i^R as a factor.

For example, if $S=\{aabb, aaac, abbb\}$, then a solution of SCS-R for S is $caaabbb$. Notice that a shortest superstring with reversals can be much shorter than a classical shortest superstring. An extremal example is given by an input set of the form $S=\{ab^h, cb^h\}$.

The SCS-R problem was already considered by Jiang et al. [9], who observed (not giving any proof) that the problem is still NP-hard. We provide a proof at the end of the paper.

In [9], the authors proposed a greedy 4-approximation algorithm. Here, we show that an adaptation of the classical greedy algorithm can be used for solving the SCS-R problem with an (optimal) compression ratio $\frac{1}{2}$, and that this algorithm can be implemented in linear time with respect to the total size of the input set.

2. Basics and notation

Let Σ be a finite alphabet. We assume that Σ is linearly sortable, e.g., $\Sigma=\{0,\ldots,n^{\mathcal{O}(1)}\}$. The *length* of a string s over Σ is denoted by $|s|$. The *empty string*, denoted by ε, is the unique string of length zero. A string t *occurs* in a string s if $s=vtz$ for some strings v,z. In this case we say that t is a *factor* of s. In particular, we say that t is a *prefix* of s when $v=\varepsilon$ and a *suffix* of s when $z=\varepsilon$. We say that a factor t is *proper* if $s \neq t$.

The string s^R obtained by reading s from right to left is called the *reversal* (or *mirror image*) of s. Given a set of strings $S=\{s_1,\ldots,s_m\}$, we define the set $S^R=\{s_1^R,\ldots,s_m^R\}$ and the set $\widetilde{S}=S\cup S^R$.

Given two strings u, v, we define the (maximum) overlap between u and v, denoted by $ov(u,v)$, as the length of the longest suffix of u that is also a prefix of v. Sometimes we abuse the notation and also say that the suffix of u of length $ov(u,v)$ is the overlap of u and v. In general $ov(u,v)$ is not equal to $ov(v,u)$, but it is readily verified that $ov(u,v)=ov(v^R,u^R)$. Additionally, we define $pr(u,v)$ as the prefix of u obtained by removing the suffix of length $ov(u,v)$ and denote $u\otimes v = pr(u,v)v$. Note that the $\otimes$ operation is in general neither symmetric nor associative.

A set of strings S is called *factor-free* if no string in S is a factor of another string in S. We say that S is *reverse-factor-free* if there are no distinct strings $u,v\in S$ such that u is a factor of v or v^R.

Given a factor-free set of strings $S=\{s_1,\ldots,s_m\}$, the SCS problem for S is known to be equivalent to that of finding a maximum-weight Hamiltonian path π in the *overlap graph* G_S, which is a directed weighted graph (S,E,w) with arcs $E=\{(s_i,s_j)\mid i\neq j\}$ of weights $w(s_i,s_j)=ov(s_i,s_j)$ (cf. Theorem 2.3 in [15]). In this setting, a path $\pi=s_{i_1},\ldots,s_{i_k}$ corresponds to a string $str(\pi):=pr(s_{i_1},s_{i_2})\cdots pr(s_{i_{k-1}},s_{i_k})s_{i_k}$. By $ov(\pi)$ we denote the total weight of arcs in the path π.

To accommodate reversals we extend the notion of an overlap graph to $\widetilde{G}_S=(V,E,w)$. Here $V=\{v_s : s\in S\}\cup\{v_s^R : s\in S\}$ so every $s\in S$ corresponds to exactly two vertices, v_s and v_s^R. We define $str(v_s)=s$ and $str(v_s^R)=s^R$. For a vertex $\alpha\in\widetilde{G}_S$ we define α^R as v_s^R if $\alpha=v_s$ for some s or as v_s if $\alpha=v_s^R$ for some s. Note that $str(\alpha^R)=str(\alpha)^R$. For every $\alpha,\beta\in V$, $\alpha\neq\beta$, we introduce an arc from α to β with weight $ov(str(\alpha),str(\beta))$. For an arc $e=(\alpha,\beta)$ we define $e^R=(\beta^R,\alpha^R)$. Note that the weight of e^R is the same as the weight of e.

For paths π in $\widetilde{G}_S$ we also use the notions of $str(\pi)$ and $ov(\pi)$. We say that a path π in $\widetilde{G}_S$ is *semi-Hamiltonian* if π contains, for every vertex $\alpha\in\widetilde{G}_S$, exactly one of the two vertices α, α^R. Observe that a solution to SCS-R problem for a reverse-factor-free set S corresponds to a maximum-weight semi-Hamiltonian path π in the overlap graph $\widetilde{G}_S$.

3. Greedy algorithm and its linear-time implementation

We define an auxiliary procedure MAKE-REVERSE-FACTOR-FREE(S) that removes from S all strings u which are contained as a factor in v or v^R for some $v\in S$, $v\neq u$. Note that the resulting set S' is reverse-factor-free and, moreover, a string is a common superstring with reversals for S' if and only if it is a common superstring with reversals for S.

Example 1. Let $S=\{ab, aaa, aab, baa\}$. Then MAKE-REVERSE-FACTOR-FREE(S) produces $S'=\{aaa, aab\}$ or $S'=\{aaa, baa\}$.

Download English Version:

https://daneshyari.com/en/article/427067

Download Persian Version:

https://daneshyari.com/article/427067

Daneshyari.com