Shortest Superstring

Presented by Nan Li

November 16, 2010

Outline

Problem Description

Given a finite alphabet Σ, and a set of n strings, $S = \{s_1, ..., s_n\} \subseteq \Sigma^+$, find a shortest string s that contains each s_i as a substring. Without loss of generality, we may assume that no string s_i is a substring of another string s_j, $j \neq i$.

Applications

- Data Compression
- Sparse Matrix Compression
- Computational Biology
- DNA-Sequencing
- Shortest Test Paths

The Initial Algorithm

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 obtain a single string.

Observation on the algorithm

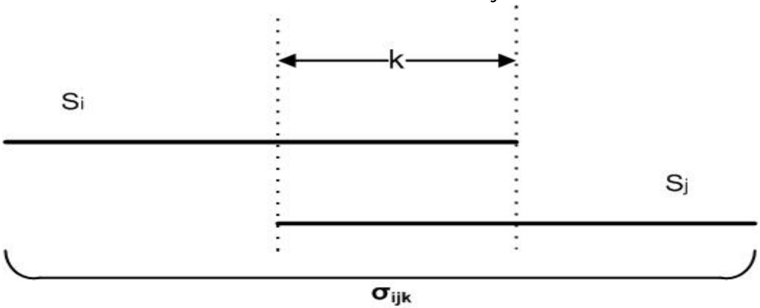
Consider an input consisting of 3 strings: ab^k, b^kc, and b^{k+1}. If the first two strings are selected in the first iteration, the greedy algorithm produces the string ab^kcb^{k+1}. This is almost twice as long as the shortest superstring, $ab^{k+1}c$.

Conversion

- Construct a set cover
- Use the greedy set cover algorithm

Set Cover Construction

For $s_i, s_j \in S$ and $k > 0$, if the last k symbols of s_i are the same as the first k symbols of s_j, let σ_{ijk} be the string obtained by overlapping these k positions of s_i and s_j



Set Cover Construction Cont.

Let M be the set that consists of the string σ_{ijk}, for all valid choices of i, j, k. For a string $\pi \in \Sigma^+$, define $\text{set}(\pi) = \{ \text{s} \in \text{S} \mid \text{s}$ is a substring of π $\}$. The universal set of the set cover instance SC is S, and the specified subsets of S are $\text{set}(\pi)$, for each string $\pi \in S \cup M$. The cost of $\text{set}(\pi)$ is $\mid \pi \mid$, i.e., the length of string π.

Shortest superstring via set cover

- Use the greedy set cover algorithm to find a cover for the instance SC. Let set(π_1), ... , set(π_k) be the sets picked by this cover.
- Concatenate the strings $\pi_1, \ldots, \pi_k$, in any order.
- Output the resulting string, say s.

An example

$\Sigma = \{0, 1\}$, $S = \{ s_1 = 001, s_2 = 01101, s_3 = 010\}$.
$M = \{ \sigma_{12} = 001101, \sigma_{13} = 0010, \sigma_{21} = \emptyset , \sigma_{23} = 011010, \sigma_{31} = 01001, \sigma_{32} = 0101101 \}$.
For the set cover instance $SC(X, F)$, $X = S$; $F = S \cup M$.
The cost-effectiveness of S: $c(S)/ \mid S - C \mid$
In the first iteration, we pick $\sigma_{13} = 0010$;
In the second iteration, we pick $s_2 = 01101$;

Analysis

Lemma

$OPT \leq OPT_{SC} \leq 2 \cdot OPT$

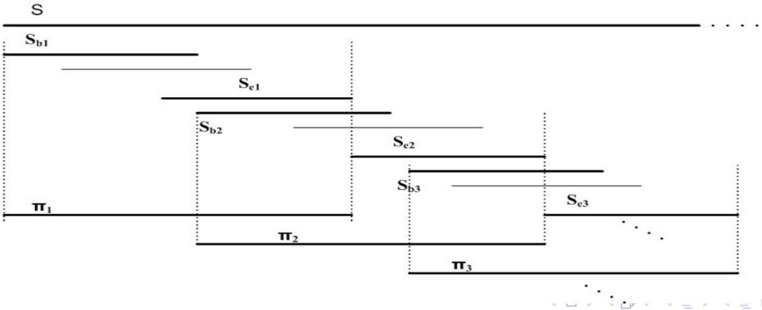
Proof.

- Consider an optimal set cover, say $\{\text{ set } (\pi_{i_j}) \mid 1 \leq j \leq l\}$, and obtain a string s by concatenating the strings $\pi_{i_j}, 1 \leq j \leq l$, in any order. $\mid s \mid = OPT_{SC}$.
- each string of S is a substring of some $\pi_{i_j}, 1 \leq j \leq l$. Hence $OPT_{SC} = \mid s \mid \geq OPT$.

□

Analysis Cont.

- Consider the leftmost occurrence of the strings $s_1, ..., s_n$ in string s.
- partition the ordered list of strings $s_1, ..., s_n$ in groups
- π_i does not overlap π_{i+2}



Analysis Cont.

Theorem

The algorithm is a $2H_n$ factor algorithm for the shortest superstring problem, where n is the number of strings in the given instance.

Proof.

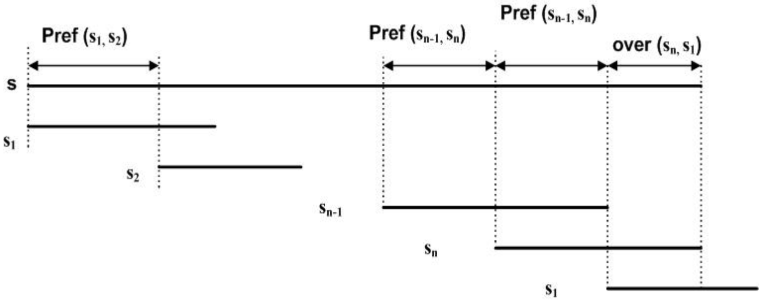
- Strings $\rightarrow$ set covers: 2 approximation
- The greedy algorithm is no better than $O(\ln n)$ approximation

□

The factor 4 Algorithm

Find another algorithm that achieves an approximation factor of 4 for the shortest superstring problem

The Idea



$OPT = \mid prefix(s_1, s_2) \mid + \mid prefix(s_2, s_3) \mid + ... + \mid prefix(s_n, s_1) \mid + \mid overlap(s_n, s_1) \mid$

Prefix graph of S

- A directed graph on vertex set $\{1, ..., n\}$
- Vertices are the corresponding strings
- An edge $i \rightarrow j$ of weight $|\ prefix(s_i, s_j)\ |$ for each $i, j, i \neq j$
- the minimum weight of a traveling salesman tour of the prefix graph gives a lower bound on OPT

Minimum weight of a cycle cover of the prefix graph

- A cycle cover is a collection of disjoint cycles covering all vertices
- Construct the following bipartite graph, $H.U = \{u_1, ..., u_n\}$ and $V = \{v_1, ..., v_n\}$ are the vertex sets of the two sides of the bipartition.
- For each $i, j \in \{v_1, ..., v_n\}$, add edge (u_i, v_j) of weight $\mid prefix(s_i, s_j) \mid$.
- find a minimum weight cycle cover reduces to finding a minimum weight perfect matching in H

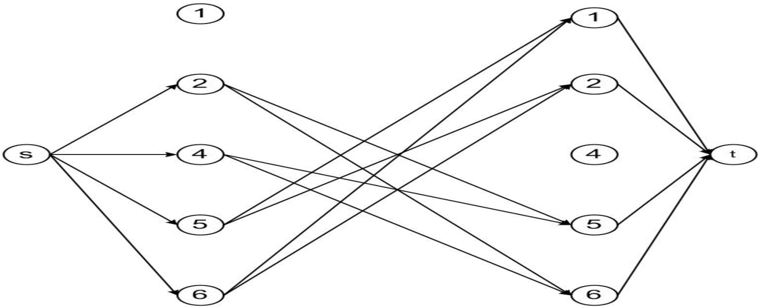
The Algorithm

- Construct the prefix graph corresponding to strings in S
- Find a minimum weight cycle cover of the prefix graph, $C = \{c_1, ..., c_k\}$
- Output $\sigma(c_1) \circ ... \circ \sigma(c_k)$

An example

A graph: 1. [2, ,3, 1, 5, 6]; 2. [1, 2, 3, 1]; 3. [0, 1, 5, 6]; 4. [0, 1, 2, 3]; 5. [2, 3, 1, 2]; 6. [3, 1, 2, 3]
The prefix graph: (2, 5), (2, 6), (4, 5), (4 ,6), (5 ,1), (5, 2), (6, 1), (6, 2). . .

An example



An example

- The maximum matchings:
  (2, 5), (4, 6), (5, 1), (6, 2)
  (2, 6), (4, 5), (5, 1), (6, 2)
  (2, 5), (4, 6), (5, 2), (6, 1)
  (2, 6), (4, 5), (5, 2), (6, 1)
- The minimum cycle cover: 4, 6, 2, 5, 1

Analysis

Lemma

If each stirng in $S' \subseteq S$ is a substring of t^{∞} for a string t, then there is a cycle of weight at most $\mid t \mid$ in the prefix graph covering all the vertices corresponding to strings in S'.

Proof.

- For each string in S', locate the starting point of its first occurrence in t^{∞}
- All these starting points will be distinct and will be lie in the first copy of t
- The weight of this cycle is at most $\mid t \mid$

□

Analysis Cont.

Lemma

Let c and c' be two cycles in C, and let r, r' be representative strings from these cycles. Then $\mid overlap(r, r') \mid < wt(c) + wt(c')$. $wt(c)$ is the weight of cycle c, i.e. $\mid prefix(s_1, s_2) \circ ... \circ prefix(s_n, s_1) \mid$, if $c = (s_1 \rightarrow s_2 ... s_n \rightarrow s_1)$.

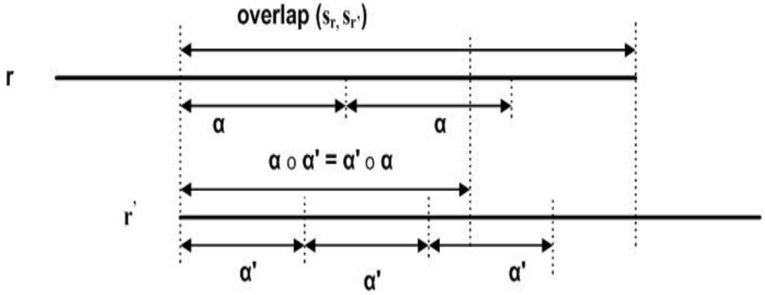
Proof.

- Consider the contradiction, $\mid overlap(r, r') \mid \geq wt(c) + wt(c')$
- α the prefix of length $wt(c)$ of $ovelap(r, r')$
- $\alpha^k \circ (\alpha')^k = (\alpha')^k \circ \alpha^k$

□

Analysis Cont.

Proof.



□

Analysis Cont.

Theorem

The Algorithm achieves an approximation factor of 4 for the shortest superstring problem

Proof.

- Let $wt(C) = \sum_{i=1}^{k} wt(c_i)$. The output of the algorithm has length

$$\sum_{i=1}^{k} \mid \sigma(c_i) \mid = wt(C) + \sum_{i=1}^{k} \mid r_i \mid$$

□

Analysis Cont.

Proof.

- $$OPT \geq \sum_{i=1}^{k} \mid r_i \mid - \sum_{i=1}^{k-1} \mid overlap(r_i, r_{i+1}) \mid \geq \sum_{i=1}^{k} \mid r_i \mid -2 \sum_{i=1}^{k} \mid wt(c_i)$$

- $$\sum_{i=1}^{k} \mid r_i \mid \leq OPT + 2 \sum_{i=1}^{k} \mid wt(c_i) \mid \leq 3 \cdot OPT$$

□

References

- V. V. Vazirani. *Approximation Algorithm*. Springer, 2003.
- Valika K. Wan and Khanh Do Ba. Set Cover and Application to Shortest Superstring. 2005
- Dr. Fei Li's slides