Lecture 4: The Shortest Superstring Problem

In this lecture we describe a 4-approximation algorithm for the shortest superstring problem.

References: Vazirani, Chapter 7 and 16.17.1 in Gusfield, *Algorithms on Strings, Trees and Sequences*.

A $(2 \ln n)$-approximation based on the Set-Cover algorithm we learnt can be found in Vazirani, Chapter 2.

The Shortest Superstring problem

Definition: Given a finite alphabet Σ and a set of n strings $S = \{s_1, \ldots, s_n\} \subseteq \Sigma^+$ a *superstring* of S is a string s that contains all of the s_i. *Find a shortest string* s. We assume that no string in S is a substring of any other.

Motivated by problems in DNA and data compression. Problem is NP-Hard.

Example: $S = \{\text{abcc, efab, bccla}\}$.
Both `bcclabccefab` and `efabccla` are superstrings of S.
`efabccla` is a shortest superstring.

For two strings s, s' let *overlap*(s, s') be the maximum overlap between a suffix of s and a prefix of s'. Let *prefix*(s, s') be the prefix of s that remains after chopping off *overlap*(s, s').

Example: $s = \text{GREAT}$, $s' = \text{EATEN}$. Then

$$\mathit{overlap}(s, s') = \text{EAT} \qquad \mathit{prefix}(s, s') = \text{GR}.$$

Now suppose that in the optimum solution the strings appear, from left to right, in the order $s_1, s_2, \ldots, s_n$. The overlap between two consecutive strings is as large as possible (otherwise a smaller superstring exists). Therefore:

$$\begin{aligned} OPT \;=\; & |\mathit{prefix}(s_1, s_2)| + |\mathit{prefix}(s_2, s_3)| + \cdots + |\mathit{prefix}(s_{n-1}, s_n)| \\ & + |\mathit{prefix}(s_n, s_1)| + |\mathit{overlap}(s_n, s_1)|. \end{aligned}$$

The Prefix Graph

Let $S = \{s_1, \ldots, s_n\} \subseteq \Sigma^+$ be as described. Its *prefix graph* is a complete weighted directed graph $G = (V, E)$ with

$$\begin{aligned} V &= \{1, 2, \ldots, n\} \\ wt(i, j) &= |\mathit{prefix}(s_i, s_j)|. \end{aligned}$$

Notice that the graph tour $i_1 \to i_2 \to \ldots \to i_n \to i_1$ has cost

$$|\mathit{prefix}(s_{i_1}, s_{i_2})| + |\mathit{prefix}(s_{i_2}, s_{i_3})| + \cdots + |\mathit{prefix}(s_{i_{n-1}}, s_{i_n})| + |\mathit{prefix}(s_{i_n}, s_{i_1})|$$

so the Minimum Travelling Salesman Tour cost (TSP) in this graph is actually a lower bound on

$$\begin{aligned} OPT \;=\; & |\mathit{prefix}(s_1, s_2)| + |\mathit{prefix}(s_2, s_3)| + \cdots + |\mathit{prefix}(s_{n-1}, s_n)| \\ & + |\mathit{prefix}(s_n, s_1)| + |\mathit{overlap}(s_n, s_1)|. \end{aligned}$$

TSP is hard to use (and calculate) so we will actually use the related problem of minimum *cycle cover* (MCC).

Cycle Covers

A *Cycle Cover* of a directed graph is a disjoint set of cycles covering all of the vertices in the graph. A minimum cycle cover of a weighted directed graph is a cycle cover of minimum cost.

Note that a tour is a cycle cover so $MCC \le TSP$.
In our case this means that $MCC \le TSP \le OPT$, giving another lower bound on OPT.

Let $G = (V, E)$ be a directed graph with $V = \{v_1, v_2, \ldots v_n\}$. Now construct a new bipartite graph $G' = (U \cup W, E')$ with $U = \{u_1, u_2, \ldots u_n\}$ and $W = \{w_1, w_2, \ldots w_n\}$ as follows:

$$(u_i, w_j) \in E' \qquad \Leftrightarrow \qquad (v_i, v_j) \in E.$$

Also, let $wt(u_i, w_j) = c(v_i, v_j)$.

Every cycle cover in G corresponds to a perfect matching in G' with the same cost and vice versa.

To find a minimum cost cycle cover in G all we have to do is find a minimum cost perfect matching in G', which can be done in polynomial time.

If $c = i_1 \to i_2 \to \ldots \to i_l \to i_1$ is a cycle in the prefix graph define

$$\begin{aligned}\alpha(c) &= \mathit{prefix}(s_{i_1}, s_{i_2}) \circ \ldots \circ \mathit{prefix}(s_{i_{l-1}}, s_{i_l}) \circ \mathit{prefix}(s_{i_l}, s_{i_1}) \\ \sigma(c) &= \alpha(c) \circ s_{i_1}.\end{aligned}$$

Notice that

- $|\alpha(c)| = wt(c)$, $|\sigma(c)| = wt(c) + |s_{i_1}|$.
- $\sigma(c)$ is a superstring of $s_{i_1}, s_{i_2}, \ldots, s_{i_l}$.
- All of the $s_{i_1}, s_{i_2}, \ldots, s_{i_l}$ are substrings of $\alpha(c) \circ \alpha(c) \circ \alpha(c) \circ \ldots = (\alpha(c))^\infty$.
- $\sigma(c)$ was constructed by *opening* cycle c at arbitrary string s_{i_1}. We call s_{i_1} the *representative string* for c.

Shortest Superstring
1. Construct the prefix graph of S.
2. Find a minimal cycle cover of the prefix graph $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$.
3. Output $\sigma(c_1) \circ \sigma(c_2) \circ \ldots \circ \sigma(c_k)$.

Theorem: SS is a 4-approximation algorithm for the shortest superstring problem.

Overlap Lemma: Let c and c' be cycles in $\mathcal{C}$ and r, r' representative strings. Then

$$overlap(r, r') < wt(c) + wt(c')$$

where $wt(c)$ is the cost of cycle c.

Theorem: SS is a 4-approximation algorithm for the shortest superstring problem.

Proof: Recall that SS finds a cycle cover $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ and outputs

$$\sigma(c_1) \circ \sigma(c_2) \circ \ldots \circ \sigma(c_k)$$

where $\sigma(c_i) = \alpha(c_1) \circ r_i$ and $|\sigma(c)| = wt(c) + |r_i|$. Let $wt(\mathcal{C}) = \Sigma_{i=1}^{k} wt(c_i)$. Then the output of the algorithm has length

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

Recall that $wt(\mathcal{C}) \leq TSP \leq OPT$. To prove the theorem we only have to prove that $\Sigma_{i=1}^{k} |r_i| \leq 3OPT$. Assume that $r_1, r_2, \ldots, r_k$ are numbered in order of their leftmost appearance in the shortest superstring of S. Then

$$OPT = \sum_{i=1}^{k} |r_i| - \sum_{i=1}^{k-1} overlap(r_i, r_{i+1}).$$

From the overlap lemma:

$$overlap(r_i, r_{i+1}) < wt(c_i) + wt(c_{i+1}).$$

Thus

$$OPT \geq \sum_{i=1}^{k} |r_i| - 2 \sum_{i=1}^{k} wt(c_i)$$

so

$$\sum_{i=1}^{k} |r_i| \leq OPT + 2wt(\mathcal{C}) \leq 3OPT$$

and we are done.