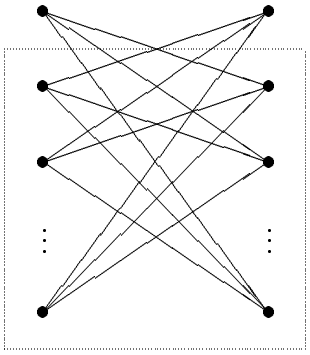
a vertex with very high weight can be placed on every edge.):



Assuming that the graph is cyclomatic weighted, each vertex receives the same weight. The decomposition obtained by the algorithm consists of only one non-trivial graph, G itself, on which the algorithm computes a minimal feedback vertex set. A possible output of the algorithm is the set shown above; this set contains $2n-2$ vertices as compared with the optimum of n given by one side of the bipartition. □

Remark: We can reduce the weighted vertex cover problem to the minimum feedback vertex set problem by duplicating every edge and then placing a very high weight vertex on each edge. Hence, any improvement to the approximation factor for the minimum feedback vertex set problem will carry over to the weighted vertex cover problem.

Exercise 6.11 A natural greedy algorithm for finding a minimum feedback vertex set is to repeatedly pick and remove the most cost-effective vertex, i.e., a vertex minimizing $\frac{w(v)}{\delta_H(v)}$, where H is the current graph, until there are no more cycles left. Give examples to show that this is not a constant factor algorithm. What is the approximation guarantee of this algorithm?

Exercise 6.12 Obtain a factor 2 approximation algorithm for the weighted vertex cover problem using the technique of layering. Use the following obvious fact to construct the layering: if the weight of each vertex is equal to its degree in G, i.e., the graph is *degree weighted*, then the weight of any vertex cover is within twice the optimal.

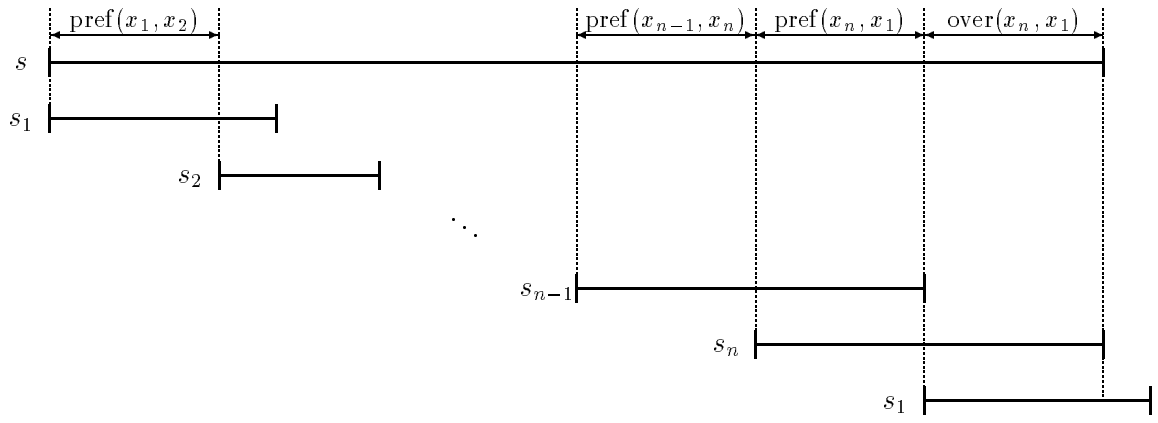
Chapter 7

Shortest superstring

In Chapter 2 we defined the shortest superstring problem and gave a $2 \cdot H_n$ factor algorithm for it by reducing to the set cover problem. In this chapter, we will first give a factor 4 algorithm, and then we will improve this to factor 3.

A factor 4 algorithm

We begin by developing a good lower bound on OPT. Let us assume that $s_1, s_2, \ldots, s_n$ are numbered in order of leftmost occurrence in the shortest superstring, s.



Let overlap(s_i, s_j) denote the maximum overlap between s_i and s_j, i.e., the longest suffix of s_i that is a prefix of s_j. Also, let prefix(s_i, s_j) be the prefix of s_i obtained by removing its overlap with s_j. The overlap in s between two consecutive s_i's is maximum possible, because otherwise a shorter superstring can be obtained. Hence, assuming that no s_i is substring of another, we get

$$\text{OPT} = |\text{prefix}(s_1, s_2)| + |\text{prefix}(s_2, s_3)| + \ldots + |prefix(s_n, s_1)| + |\text{overlap}(s_n, s_1)|. \qquad (7.1)$$

Notice that we have repeated s_1 in the end in order to obtain the last two terms of (7.1). This equality shows the close relation between the shortest superstring of S and the minimum traveling salesman tour on the *prefix graph of* S, defined as the directed graph on vertex set $\{1, \ldots, n\}$ that contains an edge $i \to j$ of weight $|\text{prefix}(s_i, s_j)|$ for each i, j (self loops included). Clearly, $|\text{prefix}(s_1, s_2)| + |\text{prefix}(s_2, s_3)| + \ldots + |prefix(s_n, s_1)|$ represents the weight of the tour

$1 \to 2 \to \ldots \to n \to 1$. Hence, by (7.1), the minimum weight of a traveling salesman tour of the prefix graph gives a lower bound on OPT. As such, this lower bound is not very useful, since we cannot efficiently compute a minimum traveling salesman tour.

The key idea is to lower bound OPT using the minimum weight of a *cycle cover* of the prefix graph (a cycle cover is a collection of disjoint cycles covering all vertices). Since the tour $1 \to 2 \to \ldots \to n \to 1$ is a cycle cover, from (7.1) we get that the minimum weight of a cycle cover lower bounds OPT.

Unlike minimum TSP, a minimum weight cycle cover can be computed in polynomial time. We first construct a bipartite version of the prefix graph: let $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$ be the two sides of the bipartition, for each $i, j \in \{1, \ldots, n\}$ add an edge of weight $|\text{prefix}(s_i, s_j)|$ from u_i to v_j. It is easy to see that each cycle cover of the prefix graph corresponds to a perfect matching of the same weight in this bipartite graph, and vice versa. Hence, finding a minimum weight cycle cover reduces to finding a minimum weight perfect matching in the bipartite graph.

If $c = (i_1 \to i_2 \to \ldots i_l \to i_1)$ is a cycle in the prefix graph, let

$$\alpha(c) = \text{prefix}(s_{i_1}, s_{i_2}) \circ \ldots \circ \text{prefix}(s_{i_{l-1}}, s_{i_l}) \circ \text{prefix}(s_{i_l}, s_{i_1}).$$

Notice that each string $s_{i_1}, s_{i_2}, \cdots s_{i_l}$ is a substring of $(\alpha(c))^\infty$. Next, let

$$\sigma(c) = \alpha(c) \circ s_{i_1}.$$

Then $\sigma(c)$ is a superstring of $s_{i_1}, \ldots, s_{i_l}$[1]. In the above construction, we "opened" cycle c at an arbitrary string s_{i_1}. For the rest of the algorithm, we will call s_{i_1} the *representative string* for c. We can now state the complete algorithm:

Algorithm 7.1 (Shortest superstring – factor 4)

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

Clearly, the output is a superstring of the strings in S. Notice that if in each of the cycles we can find a representative string of length at most the weight of the cycle, then the string output is within $2 \cdot \text{OPT}$. So, the hard case is when all strings of some cycle c are long. But since they must all be substrings of $(\alpha(c))^\infty$, they must be periodic. This will be used to prove Lemma 7.4, which establishes another lower bound on OPT. This in turn will give:

Theorem 7.2 *Algorithm 7.1 achieves an approximation factor of 4 for the shortest superstring problem.*

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

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

[1] This remains true even for the shorter string $\alpha(c) \circ \text{overlap}(s_l, s_1)$. We will work with $\sigma(c)$, since it will be needed for the factor 3 algorithm presented in next section, where we use the property that $\sigma(c)$ begins and ends with a copy of s_{i_1}.

where r_i denotes the representative string from cycle c_i. We have shown that $\mathrm{wt}(\mathcal{C}) \leq \mathrm{OPT}$. Next, we show that the sum of the lengths of representative strings is at most $3 \cdot \mathrm{OPT}$.

Assume that $r_1, \ldots, r_k$ are numbered in order of their leftmost occurrence in the shortest superstring of S. Using Lemma 7.4, we get the following lower bound on OPT:

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

Hence,

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

□

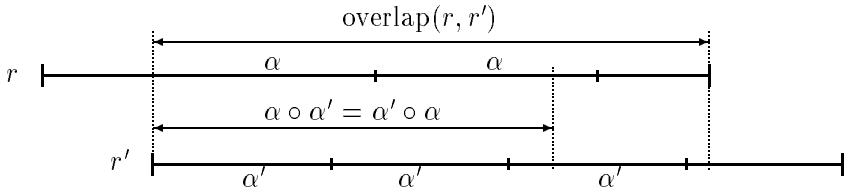
Lemma 7.3 *If each string in $S' \subseteq S$ is a substring of t^∞ for a string t, then there is a cycle of weight at most $|t|$ in the prefix graph covering all the vertices corresponding to strings in S'.*

Proof : For each string in S', locate the starting point of its first occurrence in t^∞. Clearly, all these starting points will be distinct (since no string in S is a substring of another) and will lie in the first copy of t. Consider the cycle in the prefix graph visiting the corresponding vertices in this order. Clearly, the weight of this cycle is at most $|t|$. □

Lemma 7.4 *Let c and c' be two cycles in $\mathcal{C}$, and let r, r' be representative strings from these cycles. Then*

$$|\mathrm{overlap}(r, r')| < \mathrm{wt}(c) + \mathrm{wt}(c').$$

Proof : Suppose, for contradiction, that $|\mathrm{overlap}(r, r')| \geq \mathrm{wt}(c) + \mathrm{wt}(c')$. Denote by α (α') the prefix of length $\mathrm{wt}(c)$ ($\mathrm{wt}(c')$, respectively) of $\mathrm{overlap}(r, r')$.



Clearly, $\mathrm{overlap}(r, r')$ is a prefix of both α^∞ and $(\alpha')^\infty$. In addition, α is a prefix of $(\alpha')^\infty$ and α' is a prefix of α^∞. Since $\mathrm{overlap}(r, r') \geq |\alpha| + |\alpha'|$, it follows that α and α' commute, i.e., $\alpha \circ \alpha' = \alpha' \circ \alpha$. Bur then, $\alpha^\infty = (\alpha')^\infty$. This is so because for any $k > 0$,

$$\alpha^k \circ (\alpha')^k = (\alpha')^k \circ \alpha^k.$$

Hence, for any $N > 0$, the prefix of length N of α^∞ is the same as that of $(\alpha')^\infty$.

Now, by Lemma 7.3, there is a cycle of weight at most $\mathrm{wt}(c)$ in the prefix graph covering all strings in c and c', contradicting the fact that $\mathcal{C}$ is a minimum weight cycle cover. □

Exercise 7.5 Show that Lemma 7.4 cannot be strengthened to

$$|\text{overlap}(r, r')| < \max\{\text{wt}(c), \text{wt}(c')\}.$$

Improving to factor 3

Notice that any superstring of the strings $\sigma(c_i)$, $i = 1, \ldots, k$, is also a superstring of all strings in S. So, instead of simply concatenating these strings, let us make them overlap as much as possible (this may sound circular, but it is not!).

Let us define the *compression* achieved by a superstring as the difference between the total length of the input strings and the length of the resulting superstring. Clearly, maximum compression is achieved by the shortest superstring. Several algorithms are known to achieve at least half the optimal compression. For instance, the greedy superstring algorithm presented in a previous lecture does so, though its proof is based on a complicated case analysis. A less efficient algorithm, based on cycle cover, is presented at the end.

Algorithm 7.6 (Shortest superstring – factor 3)

1. Construct the prefix graph corresponding to strings in S.
2. Find a minimum cycle cover of the prefix graph, $\mathcal{C} = \{c_1, \ldots, c_k\}$.
3. Run the greedy algorithm on $\{\sigma(c_1), \ldots, \sigma(c_k)\}$ and output the resulting string, $\overline{\sigma}$.

Let OPT_σ denote the length of the shortest superstring of the strings in $S_\sigma = \{\sigma(c_1) \ldots \sigma(c_k)\}$, and let r_i be the representative string of c_i.

Lemma 7.7 $|\overline{\sigma}| \leq \text{OPT}_\sigma + \text{wt}(\mathcal{C})$.

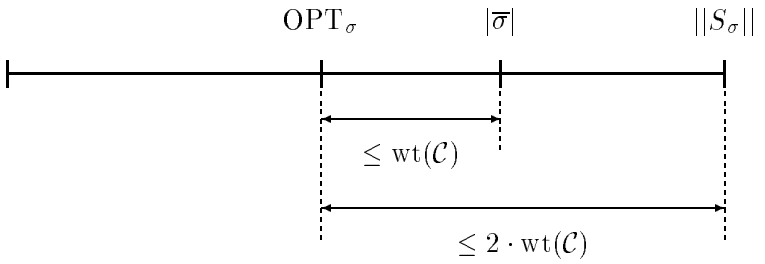
Proof : Assuming that $\sigma(c_1), \ldots, \sigma(c_k)$ appear in this order in a shortest superstring of S_σ, the maximum compression that can be achieved on S_σ is given by

$$\sum_{i=1}^{k-1} |\text{overlap}(\sigma(c_i), \sigma(c_{i+1}))|.$$

Since each string $\sigma(c_i)$ has r_i as a prefix as well as suffix, by Lemma 7.4,

$$|\text{overlap}(\sigma(c_i), \sigma(c_{i+1}))| \leq \text{wt}(c_i) + \text{wt}(c_{i+1}).$$

Hence, the maximum compression achievable on S_σ is at most $2 \cdot \text{wt}(\mathcal{C})$, i.e., $||S_\sigma|| - \text{OPT}_\sigma \leq 2 \cdot \text{wt}(\mathcal{C})$, where $||X||$ denotes the sum of the lengths of the strings in X.



The compression achieved by the greedy algorithm on S_σ is at least half the maximum compression, hence $|\overline{\sigma}|$ is closer to OPT_σ than to $||S_\sigma||$, and the lemma follows. □

Finally, we relate OPT_σ to OPT.

Lemma 7.8 $\mathrm{OPT}_\sigma \leq \mathrm{OPT} + \mathrm{wt}(\mathcal{C})$.

Proof: Let OPT_r denote the length of the shortest superstring of the strings in $S_r = \{r_1, \ldots, r_k\}$. Because each $\sigma(c_i)$ begins and ends with r_i, the compression achievable on the strings of S_σ is at least as large as that achievable on the strings of S_r. Thus,

$$||S_\sigma|| - \mathrm{OPT}_\sigma \geq ||S_r|| - \mathrm{OPT}_r.$$

Clearly, $||S_\sigma|| = ||S_r|| + \mathrm{wt}(\mathcal{C})$. This gives

$$\mathrm{OPT}_\sigma \leq \mathrm{OPT}_r + \mathrm{wt}(\mathcal{C}).$$

The lemma follows by noticing that $\mathrm{OPT}_r \leq \mathrm{OPT}$. □

Combining the previous two lemmas we get:

Theorem 7.9 *Algorithm 7.6 achieves an approximation factor of 3 for the shortest superstring problem.*

Finally, we present an algorithm achieving at least half the optimal compression. Suppose that the strings to be compressed, $s_1, \cdots, s_k$, are numbered in the order in which they appear in a shortest superstring. Then, the optimal compression is given by

$$\sum_{i=1}^{k-1} |\mathrm{overlap}(\sigma_i, \sigma_{i+1})|.$$

This is the weight of the traveling salesman path $1 \to 2 \to \ldots \to k$ in the *overlap graph* of the strings $s_1, \cdots, s_k$; this graph contains an arc $i \to j$ of weight $|\mathrm{overlap}(s_i, s_j)|$ for each $i \neq j$ (thus this graph has no self loops). Hence, the optimal compression is upper bounded by the maximum traveling salesman tour in the overlap graph, which in turn is upper bounded by the maximum cycle cover. The latter can be computed in polynomial time using matching, similar to a minimum weight cycle cover. the associated bipartite graph and finding a maximum weight perfect matching in it. Since the overlap graph has no self loops, each cycle has length at least 2. So, the lightest

edge in each cycle has weight at most half that of the cycle. Open each cycle by discarding the lightest edge, overlap strings in the order given by remaining paths, and concatenate the resulting strings. This gives a superstring achieving compression of at least half the weight of the cycle cover, hence also half the optimal compression.