

The Minimum Substring Cover Problem

Danny Hermelin^{*1}, Dror Rawitz^{**2},
Romeo Rizzi^{***3}, and Stéphane Vialette^{***4}

¹ Department of Computer Science,
University of Haifa, Haifa 31905, Israel
`danny@cri.haifa.ac.il`

² School of Electrical Engineering,
Tel-Aviv University, Tel-Aviv 69978, Israel
`rawitz@eng.tau.ac.il`

³ Dipartimento di Matematica ed Informatica (DIMI),
Università di Udine, I-33100 Udine, Italy
`Romeo.Rizzi@dimi.uniud.it`

⁴ Laboratoire de Recherche en Informatique (LRI),
Université Paris-Sud, 91405 Orsay, France
`viallette@lri.fr`

Abstract. In this paper we consider the problem of covering a set of strings S with a set C of substrings in S , where C is said to cover S if every string in S can be written as a concatenation of the substrings in C . We discuss applications for the problem that arise in the context of computational biology and formal language theory. We then proceed to show that this problem is at least as hard as the MINIMUM SET COVER problem. In the main part of the paper, we focus on devising approximation algorithms for the problem using two generic paradigms – the local-ratio technique and linear programming rounding.

1 Introduction

In a *covering problem* we are faced with the following situation: We are given two (not necessarily disjoint) sets of elements, the *base elements* and the *covering elements*, and the goal is to find a minimum (weight) subset of covering elements that “covers” all the base elements. The exact notion of covering differs from problem to problem, yet this abstract setting is common to many classical combinatorial problems in various application areas. Two famous examples are MINIMUM SET COVER – where the covering elements are subsets of the base elements and the notion of covering corresponds to set inclusion – and MINIMUM VERTEX COVER – where the setting is graph-theoretic and the notion of covering corresponds to incidence between vertices and edges. Ever since the early days of combinatorial optimization, research on covering problems such as the two examples above proved extremely fruitful in laying down fundamental techniques and ideas. The early work of Johnson [13] and Lovász [14] on MINIMUM SET COVER pioneered the greedy analysis approach, while Chvátal [7] gave the first analysis based on linear programming (LP) while tackling the same problem. The first LP-rounding algorithm by Hochbaum [12] was also designed for MINIMUM SET COVER, while Bar-Yehuda and Even gave the first Primal-Dual [3] and Local-Ratio [4] algorithms for MINIMUM VERTEX COVER.

In this paper we introduce a new covering problem which resides in the realm of strings. A string c is a *substring* of a string s , if c can be obtained by deleting any number of consecutive

* Partially supported by the Caesarea Rothschild Institute.

** Most of the work on this paper was done while the second author was a postdoc at the Caesarea Rothschild Institute, University of Haifa.

*** Supported by the Italian-French PAI Galileo Project 08484VH.

letters from both ends of s . In our covering problem, the base elements are strings and the covering elements are their substrings. The notion of covering corresponds to string-factorization, or to the generation of strings by substring concatenation. More formally, for a given set of strings S , let $\mathcal{C}(S)$ denote the set of all substrings of strings in S . We define a *cover* of S to be a subset $C \subseteq \mathcal{C}(S)$ such that any string $s \in S$ can be written as a concatenation of strings in C . If each string in S can be written as a concatenation of at most ℓ strings in C , we say that C is an ℓ -cover of S . Given a weight function $w : \mathcal{C}(S) \rightarrow \mathbb{Q}^+$, we are interested in computing an ℓ -cover of S with minimum possible weight:

MINIMUM SUBSTRING COVER:

Instance: A set of strings S , a weight function $w : \mathcal{C}(S) \rightarrow \mathbb{Q}^+$, and an integer $\ell \geq 2$.

Solution: An ℓ -cover C of S . That is, a set of strings $C \subseteq \mathcal{C}(S)$, where for each $s \in S$ there exist $c_1, \dots, c_p \in C$, $p \leq \ell$, with $s = c_1 \cdots c_p$.

Measure: Total weight of the cover, *i.e.* $w(C) = \sum_{c \in C} w(c)$.

Example 1. Consider the set of strings $S = \{\text{'a'}, \text{'aab'}, \text{'aba'}\}$. Then $\mathcal{C}(S) = \{\text{'a'}, \text{'b'}, \text{'aa'}, \text{'ab'}, \text{'ba'}, \text{'aab'}, \text{'aba'}\}$, and $C_1 = \{\text{'a'}, \text{'b'}\}$ and $C_2 = \{\text{'a'}, \text{'ab'}\}$ are covers of S . The cover C_1 is a 3-cover of S , while C_2 is a 2-cover.

Throughout the paper, we use n to denote the number of strings in S , and m to denote the maximum length of any string in S , *i.e.* $n = |S|$ and $m = \max\{|s| : s \in S\}$.

Note that in case $\ell \geq m$, there is no actual bound on the concatenation length of the required cover, and this case is denoted by $\ell = \infty$. An ∞ -cover is referred to simply as a cover. Another interesting special case is when $\ell = 2$. In this case, we are required to cover S with a set of prefixes and suffixes in S , where a *prefix* (resp. *suffix*) of a string s is a substring of s which is obtained by removing consecutive letters only from the end (resp. beginning) of s . As we will see, these two extremal cases both give a certain amount of combinatorial leverage, and therefore deserve particular consideration. We also wish to point out that our use of general weight functions $w : \mathcal{C}(S) \rightarrow \mathbb{Q}^+$ allows for more robustness in modeling different scenarios. For instance, when w is the *unitary function*, *i.e.* $w(c) = 1$ for every $c \in \mathcal{C}(S)$, this corresponds to the situation where we want to minimize the size of a cover of S . When $w(c) = |c|$, *i.e.* the weight of every substring is its length (w is the *length-weighted function*), this corresponds to the case where we want to minimize the total length of the cover. Often some sort of middle ground between these two situations might also be desirable.

Example 2. Consider the two covers C_1 and C_2 of the set of strings S in Example 1. If w is the unitary function, then $w(C_1) = w(C_2) = 2$. However, if w is the length-weighted function, we have $w(C_1) = 2 < w(C_2) = 3$.

Our initial inspiration for studying MINIMUM SUBSTRING COVER came from a paper by Bodlaender *et al.* [5], who described an application for this problem in the context of protein folding (The authors of [5] actually referred to our problem as the DICTIONARY GENERATION problem, and considered its unweighted variant under the parameterized complexity framework.) Protein folding is the problem of determining the folding structure of proteins using their amino-acid sequential description. This problem is extremely important, since most of the functionality of a protein is determined by its folding structure, and because current biological methods for extracting the sequential description of a given protein exceed by far the methods for extracting the folding structure

of the protein. In [5], it is argued that since all known approaches for protein folding are **NP**-hard, a possible heuristic for this problem is to break the protein sequence into small segments, small enough for allowing efficient folding computation. This heuristic is justified by the fact that many proteins seem to be composed of relatively small regions which fold independently of other regions. The theory of *exon shuffling* proposes that all proteins are concatenations of such regions, where the regions are drawn from a common ancestral dictionary [9, 17].

MINIMUM SUBSTRING COVER can also model interesting computational issues which arise in formal language theory, and in particular, in the area of combinatorics of words. Our notion of cover actually corresponds to the notion of *combinatorial rank*, an important parameter of a set of words (*cf.* [6]). Neraud [15] studied the problem of determining whether a given set of words is *elementary*, where a set of strings is said to be elementary if it does not have a cover of size strictly less than its own. Neraud describes a direct application of this notion to the famous D0L-sequence equivalence problem (*cf.* [19]) via so-called elementary morphisms [10]. He also argues that this notion appears frequently in numerous sub-areas such as test sets, code theory, representation of formal languages, and the theory of equations in free monoids. His main result is in showing that deciding whether a given set of words is elementary is **coNP**-complete, which implies that MINIMUM SUBSTRING COVER is **NP**-hard.

In the remainder of this section we briefly discuss our results and the structure of the paper. In Section 2 we present some lower bounds on the approximation factors of polynomial-time algorithms for MINIMUM SUBSTRING COVER. We show that, in general, the problem is **NP**-hard to approximate within a factor of $c \ln n$ for some $c > 0$, and within $\lfloor m/2 \rfloor - 1 - \varepsilon$ and all $\varepsilon > 0$. We also show that the problem remains **APX**-hard even when m is constant, and the given weight function is either the unitary or the length-weighted function. Following this, in Section 3, we apply the local-ratio technique [2, 4] to obtain three approximation algorithms with performance ratios $\binom{m+1}{2} - 1$, $m - 1$, and m , where the last two are specializations of the first to the cases of $\ell = 2$ and $\ell = \infty$ (the latter only applies for restricted types of weight functions). Finally, we present in Section 4 an algorithm based on rounding the linear programming relaxation of the problem, which achieves a performance ratio of $\mathcal{O}(\log n \cdot m^{(\ell-1)^2/\ell})$ with high probability. This algorithm is an extension of an algorithm of Hajiaghayi *et al.* [11] used for solving the MINIMUM MULTICOLORED SUBGRAPH problem.

2 Approximation Lower Bounds

We begin our discussion by presenting some lower bounds on the performance ratios of polynomial-time approximation algorithms for MINIMUM SUBSTRING COVER. We show that in general, MINIMUM SUBSTRING COVER is **NP**-hard to approximate within factors of $c \ln n$ and $\lfloor m/2 \rfloor - 1 - \varepsilon$, for some $c > 0$ and all $\varepsilon > 0$ (recall that $n = |S|$ and $m = \max_{s \in S} |s|$). We also show that the problem is **APX**-hard even when all strings in S have length at most 4, and the given weight function $w : \mathcal{C}(S) \rightarrow \mathbb{Q}^+$ is either the unitary or the length-weighted function.

To prove our approximation lower bounds, we present an L-reduction [16] from the MINIMUM HYPERGRAPH VERTEX COVER problem, which is no more than the MINIMUM SET COVER problem when the roles of the covering and base elements are reversed. In MINIMUM HYPERGRAPH VERTEX COVER, we are given a vertex-weighted hypergraph $H = (V(H), E(H))$, $w_H : V(H) \rightarrow \mathbb{Q}^+$, and the goal is to find a minimum weight vertex cover of H . That is, a subset of vertices $V \subseteq V(H)$ of minimum weight, such that $V \cap e \neq \emptyset$ for each hyperedge $e \in E(H)$. It is known that the problem is

NP-hard to approximate within a factor of $c \ln |E(H)|$ for some constant c [18], and also **NP**-hard to approximate within $\max_{e \in E(H)} |e| - 1 - \varepsilon$ for any $\varepsilon > 0$ (assuming $\max_{e \in E(H)} |e| > 2$) [8].

Let (H, w_H) be a given instance of MINIMUM HYPERGRAPH VERTEX COVER. From (H, w_H) , we construct an instance (S, w, ℓ) for MINIMUM SUBSTRING COVER as follows. Let e_{\max} denote the largest edge of H . The set of strings S is defined over an alphabet Σ which consists of two unique letters ‘ v ’, ‘ V ’ $\in \Sigma$ for each vertex $v \in V(H)$, and an additional special unique letter ‘ $\$$ ’ $\in \Sigma$ which we use for padding. We refer to the substring ‘ vV ’ as the *encoding* of the vertex $v \in V(H)$. For each edge $e \in E(H)$, we construct a string s_e by concatenating (in any arbitrary order) the encodings of all vertices $v \in e$. In addition, we concatenate $2(|e_{\max}| - |e|)$ ‘ $\$$ ’ letters to the end of s_e . The set of strings S is defined by $S = \{s_e : e \in E(H)\}$. Note that $n = |S| = |E(H)|$, and that $m = \max_{s \in S} |S| = 2|e_{\max}|$.

Example 3. Suppose $V(H) = \{a, b, c, d\}$ and $E(H) = \{\{a, b\}, \{b, d\}, \{a, c, d\}\}$. The set of strings S is then constructed as $S = \{\text{‘aAbB\$\$’}, \text{‘bBdD\$\$’}, \text{‘aAcCdD’}\}$.

Next, we define the weight function $w : \mathcal{C}(S) \rightarrow \mathbb{Q}^+$ by

$$w(c) = \begin{cases} 0 & : c \in \Sigma, \\ w_H(v) & : c \text{ is the encoding of } v \in V(H), \\ \infty & : \text{otherwise.} \end{cases}$$

Finally, to complete the construction, we set $\ell = m - 1$.

Lemma 1. *H has a vertex cover with total weight k iff S has an ℓ -cover with total weight k .*

Proof. Suppose $V \subseteq V(H)$ is a vertex cover of H with $w_H(V) = \sum_{v \in V} w_H(v) = k$, and consider the set of substrings $C = \Sigma \cup \{\text{‘}vV\text{’} : v \in V\}$. Clearly, $w(C) = k$. Furthermore, C is an ℓ -cover of S , since C can cover any string $s_e \in S$ using $\ell - 1$ letters and a single encoding of a vertex $v \in V \cap e$.

Conversely, suppose S has an ℓ -cover C with $w(C) = k$. Write $C = C_1 \cup C_2$, where $C_1 = C \cap \Sigma$. Then $w(C_2) = k$ and C_2 consists only of substrings which are encodings of vertices in H . This is because no ℓ -cover can cover any string in S using only letters, and all non-encoding substrings of length at least 2 in $\mathcal{C}(S)$ have infinite weight. Let $V \subseteq V(H)$ be the vertices in H corresponding to the encodings in C_2 . Then $w_H(V) = w(C_2) = k$. Moreover, since C uses at least one vertex-encoding in C_2 to cover any string $s_e \in S$, it follows by our construction that $V \cap e \neq \emptyset$ for all $e \in E(H)$. \square

The lemma above implies that any α -approximation algorithm for MINIMUM SUBSTRING COVER would give an α -approximation algorithm for MINIMUM HYPERGRAPH VERTEX COVER. Hence, due to the results of [18] and [8], we can conclude that it is **NP**-hard to approximate MINIMUM SUBSTRING COVER within $c \ln n$ for some constant c , and within $\lfloor m/2 \rfloor - 1 - \varepsilon$ for any $\varepsilon > 0$. However, the construction in the lemma relies on a somewhat unnatural weight function, and on the fact that the strings in S are allowed to be fairly long. Nevertheless, we can show that a special case of this construction can be used to relax both these conditions at the cost of reducing the lower bounds to only a constant.

Consider our construction for the case where $H = G = (V(G), E(G))$ is a graph rather than a hypergraph. This special case, better known as the MINIMUM VERTEX COVER problem, is known to be **NP**-hard to approximate within some constant, even in the unweighted case, and even if each vertex in G is incident to at most three edges in $E(G)$ [1, 16]. Note that in this case, any vertex

cover of G must be of size at least $|V(G)|/4$. Consider the set of strings $\{s_e : e \in E(G)\}$ constructed as defined above. This set consists of four letter strings, each of which is a concatenation of two encodings of vertices of G (no ‘\$’ letters). Now define the input set of strings S by $S = \Sigma \cup \{s_e : e \in E(G)\}$. We can prove the following relationship between the size of a 3-cover of S and the size of a vertex cover in G .

Lemma 2. *G has a vertex cover of size k iff S has a 3-cover of size $2|V(G)| + k$.*

Proof. Suppose G has a vertex cover $V \subseteq V(G)$ of size k . Then the set of substrings $C = \Sigma \cup \{‘vV’ : v \in V\}$ is a 3-cover of S , and furthermore, $|C| = 2|V(G)| + k$.

Conversely, suppose S has a 3-cover C . Since $\Sigma \subseteq S$, C must include every letter in Σ . Hence, C is of size $2|V(G)| + k$, for some k . Let us say that C is *normalized* if it consists solely of letters and vertex-encodings. If C is normalized, we can write $C = \Sigma \cup C_1$, where C_1 is the set of $|C| - |\Sigma| = k$ vertex encodings in C , and by a similar argument used for Lemma 1, we can show that the set of k vertices $V \subseteq V(G)$ corresponding to the vertex-encodings in C_1 is a vertex cover of G . Otherwise, if C is not normalized, we can always normalize C at no cost to its total size. Indeed, note that any string $s_e \in S$ can be covered using a vertex encoding of a vertex incident to e and two additional letters in Σ . Furthermore, notice that any non-encoding substring $c \notin \Sigma$ can only be used to cover a single word in S . Hence, if C covers some string $s_e \in S$ using a non-encoding substring $c \notin \Sigma$, we can replace c with a vertex encoding of some vertex incident to e without violating the fact that C is a cover and with no increase to its total size. Doing this for all non-encoding substrings $c \in C \setminus \Sigma$, we obtain a normalized cover of S whose size is at most $|C|$. \square

Using similar arguments, we can also prove that:

Lemma 3. *G has a vertex cover of size k iff S has an 3-cover of total length $2|V(G)| + 2k$.*

Using the last two lemmas and the fact that any vertex cover of G must be of size at least $|V(G)|/4$ it is not hard to see that the above construction constitutes an L-reduction from MINIMUM VERTEX COVER on graphs with bounded degree to both unweighted MINIMUM SUBSTRING COVER and length-weighted MINIMUM SUBSTRING COVER. It follows that there is some constant c for which MINIMUM SUBSTRING COVER for constant length strings and unitary/length-weighted weight functions is **NP**-hard to approximate. Combining this with the implications of Lemma 1, we obtain the main result of this section:

Theorem 1. *MINIMUM SUBSTRING COVER is **NP**-hard to approximate*

- *within $c \ln n$ for some constant c , and within $\lfloor m/2 \rfloor - 1 - \varepsilon$ for any $\varepsilon > 0$.*
- *within some constant c , when m and ℓ are constant, and w is either the unitary or the length-weighted function.*

3 Local-Ratio Algorithms

In the previous section we gave some negative results for the MINIMUM SUBSTRING COVER problem. In this section we show how to apply the local-ratio technique [2, 4] to obtain positive results in the form of approximations algorithms with performance ratios depending on the length of the longest word in S . In particular, if m is the maximum length of any word in S , we show how to find in polynomial time an $\left(\binom{m+1}{2} - 1\right)$ -approximate ℓ -cover for S for general values of ℓ . For $\ell = 2$, we show

how to obtain $(m-1)$ -approximate covers, and for $\ell = \infty$, we show how to compute m -approximate covers. (The latter case applies only for a restricted type of weight functions.) We begin by giving a brief overview of the local-ratio technique.

The local-ratio technique [2] is based on the Local-Ratio Lemma [4], which in our terms is stated as follows:

Lemma 4 (Local-Ratio). *Let C be a cover for S , and let w_1 and w_2 be weight functions for $\mathcal{C}(S)$. If C is an α -approximate, both with respect to w_1 and with respect to w_2 , then C is also α -approximate with respect to $w_1 + w_2$.*

A local-ratio α -approximation algorithm is typically recursive and works as follows. Given a problem instance with a weight function w , we find a non-negative weight function $w_1 \leq w$ such that (1) every solution of a certain type is α -approximate with respect to w_1 , and (2) there exists some element e in our input for which $w(e) = w_1(e)$. We subtract w_1 from w and remove some zero weight element from the problem instance. Then, we recursively solve the new problem instance, while assuring that the solution returned can be fixed so that it becomes of the above mentioned type. If fixing the solution does not increase its w_1 weight, nor its $w - w_1$ weight, the Local-Ratio Lemma guarantees that this solution is α -approximate with respect to our original weight function w . The base of the recursion occurs when the problem instance has degenerated into a trivial instance.

Figure 1 gives an approximation algorithm for MINIMUM SUBSTRING COVER which is based on the local-ratio technique. We call this algorithm LR. We first show that algorithm LR computes $\binom{m+1}{2} - 1$ -approximate ℓ -covers for general values of ℓ . Following this, we show that some fine tuning of the algorithm allows us to achieve approximation ratios of $m-1$ and m , for the special cases of $\ell = 2$ and $\ell = \infty$ respectively.

Algorithm LR(S, w, ℓ)

Data : A set of strings S , a weight function $w : \mathcal{C}(S) \rightarrow \mathbb{Q}^+$, and an integer $\ell \geq 2$.
Result : An ℓ -cover C for S .
begin
 1. $C \leftarrow \{c \in \mathcal{C}(S) : w(c) = 0\}$.
 2. **if** C is an ℓ -cover of S **then return** C .
 3. Let $s \in S$ be a string not ℓ -covered by C of maximum length.
 4. $C_s \leftarrow \{c \in \mathcal{C}(S) \setminus C : c \text{ is a substring of } s\}$.
 5. Set $\varepsilon = \min\{w(c) : c \in C_s\}$.
 6. Define $w_1(c) = \begin{cases} \varepsilon & c \in C_s, \\ 0 & \text{otherwise.} \end{cases}$
 7. Define $w_2 = w - w_1$.
 8. $C \leftarrow \text{LR}(S, w_2, \ell)$.
 9. **if** $C \setminus \{s\}$ is an ℓ -cover for S **then** $C \leftarrow C \setminus \{s\}$.
 return C .
end

Fig. 1. A local ratio approximation framework.

The general outline of algorithm LR is as follows: First, the algorithm adds all substrings $c \in \mathcal{C}(S)$ with zero weight to an initial partial-solution C , since these do not have effect on the total weight of the optimal solution. Then, if C is not already a cover of S , LR selects a string

$s \in S$ not covered by C , and examines all substrings C_s of s not already in C . It then subtracts $\varepsilon = \min\{w(c) : c \in C_s\}$ from the weight of all substrings in C_s , and recurses on the new weight function. The last line of the algorithm ensures that at least one substring of s will not be included in C . Such solutions are shown to be $\binom{m+1}{2} - 1$ -approximate with respect to w_1 , and also with respect to w_2 , and therefore due to the Local-Ratio Lemma, are also $\binom{m+1}{2} - 1$ -approximate with respect to w .

Note that at each recursive call of the algorithm, at least one substring in $\mathcal{C}(S)$ which has positive weight with respect to w , will have zero weight with respect to w_2 . Hence, the algorithm is guaranteed to terminate, and furthermore, it is also guaranteed to terminate after at most polynomial-many recursive calls. It is not difficult to see that each recursive call can be carried out in polynomial-time. The only problematic line could be line 2, but this can be performed efficiently using standard dynamic-programming techniques (details omitted). Finally, observe that by its definition, algorithm LR indeed returns an ℓ -cover of S . In the following lemma we show that this cover is $\binom{m+1}{2} - 1$ -approximate.

Lemma 5. *Algorithm LR computes an $\binom{m+1}{2} - 1$ -approximate ℓ -cover of S .*

Proof. To prove that the cover C returned by algorithm LR is $\binom{m+1}{2} - 1$ -approximate, we apply induction on the number of recursive calls of the algorithm, and show that at any recursive call, C is $\binom{m+1}{2} - 1$ -approximate with respect to the given weight function w of that particular call. At the recursive basis, C has zero weight with respect to w so it is indeed $\binom{m+1}{2} - 1$ -approximate. For the inductive step, consider any recursive call other than the basis, and assume that the cover C returned at Line 8 is $\binom{m+1}{2} - 1$ -approximate with respect to w_2 . Note that C also remains $\binom{m+1}{2} - 1$ -approximate with respect to w_2 after Line 9.

Let $s \in S$ be the string selected at Line 3. Since s is of length at most m , it has at most $\binom{m+1}{2}$ distinct substrings, and so $|C_s| \leq \binom{m+1}{2}$. Hence, $\sum_{c \in \mathcal{C}(S)} w_1(c) \leq \binom{m+1}{2} \varepsilon$. Furthermore, if C includes s after Line 9, then at least one substring of s is not included in C . This is because, by our selection of s , s can only be used to cover itself among all strings not covered by zero-weight substrings in $\mathcal{C}(S)$. In any case, after Line 9 we have $C_s \not\subseteq C$, and so $|C \cap C_s| \leq \binom{m+1}{2} - 1$. Hence, $\sum_{c \in C} w_1(c) \leq (\binom{m+1}{2} - 1) \varepsilon$. Furthermore, by our selection of ε , any cover for S has weight at least ε with respect to w_1 . It follows that, after Line 9, C is $\binom{m+1}{2} - 1$ -approximate with respect to w_1 as well as with respect to w_2 . According to the Local-Ratio Lemma, the cover returned is $\binom{m+1}{2} - 1$ -approximate with respect to w , and so the lemma is proved. \square

We next show that with a small modification to algorithm LR, we can achieve an approximation factor of $m - 1$ for the special case of $\ell = 2$. First, when $\ell = 2$, we consider $\mathcal{C}(S)$ to be the set of all prefixes and suffixes of strings in S , rather than the set of all substrings of S . We use algorithm LR with the following modification. We replace Line 4 of the algorithm with:

$$C_s \leftarrow \{c \in \mathcal{C}(S) \setminus C : \exists c' \in C \text{ with } s = cc'\} \cup \{c \in \mathcal{C}(S) \setminus C : \exists c' \in \mathcal{C}(S) \text{ with } s = c'c\} \cup \{s\}.$$

That is, for every pair of prefix and suffix of s , C_s either includes the suffix if it is not already in C (*i.e.* does not have zero weight), or it includes the prefix if the suffix is already in C . Note that since $s \in C_s$, $C_s \neq \emptyset$. We denote the modified version of algorithm LR by LR₂.

It is clear that algorithm LR₂ can be implemented to run in polynomial-time. Furthermore, the analysis of the performance ratio of algorithm LR₂ is almost the same as the analysis for algorithm LR. The main difference is in the upper bound of the total w_1 weight of C . First observe

that we still have $\sum_{c \in C} w_1(c) \geq \varepsilon$ for any 2-cover C of S , since any cover must still include at least one string of C_s . On the other hand, since $|C_s| \leq m$, we have $\sum_{c \in C_s} w_1(c) \leq m \cdot \varepsilon$. Since after Line 9 we know that $C_s \not\subseteq C$, we have in fact $\sum_{c \in C_s} w_1(c) \leq (m-1) \cdot \varepsilon$.

Lemma 6. *Algorithm LR_2 computes an $(m-1)$ -approximate 2-cover of S .*

We next consider the case of $\ell = \infty$ (i.e. $\ell \geq m$). Given a weight function $w : \mathcal{C}(S) \rightarrow \mathbb{Q}^+$, we say that w is *proper* if for any $c, c_1 \in \mathcal{C}(S)$, $w(c_1) \leq w(c)$ whenever c_1 is a prefix or a suffix of c . For example, unitary and length-weighted functions are proper. We show how to modify algorithm LR so that it computes m -approximate covers for proper weight functions. Note that for length-weighted functions the problem is trivial since the solution is always the alphabet of S .

Our modified version of algorithm LR for the case of $\ell = \infty$ is called LR_∞ . It is obtained by replacing Line 9 in algorithm LR with the following line:

while $\exists c, c_1 \in C$ with $c = c_1c_2$ or $c = c_2c_1$ **do** $C \leftarrow C \setminus \{c\} \cup \{c_2\}$.

Note that this while loop requires polynomial-time because the total length of the substrings in C decreases in every iteration of the while loop. The more important observation is that, since $\ell \geq m$, C remains an ℓ -cover for S after the while loop terminates. Furthermore, after line 9, C is both prefix-free and suffix-free. That is, there are no two strings in C where one is the prefix or suffix of the other. This implies that $|C \cap C_s| \leq m$, and is precisely the property that we use to obtain our m -approximation factor.

Lemma 7. *Algorithm LR_∞ computes an m -approximate cover of S assuming the given weight function $w : \mathcal{C}(S) \rightarrow \mathbb{Q}^+$ is proper.*

Proof. First observe that if the initial weight function is proper, then all weight functions throughout the entire recursion of the algorithm are proper. This is because whenever the weight of a string decreases, the weight of all its prefixes and suffixes decreases by the same amount. Next note that Line 9 of algorithm LR_∞ does not increase the weight of C with respect to w_2 , nor with respect to w_1 , since both are proper weight functions. The approximation factor promised by the lemma is therefore obtained due to the observation that $1 \leq |C \cap C_s| \leq m$ after Line 9, and so $\varepsilon \leq \sum_{c \in C} w_1(c) \leq m \cdot \varepsilon$. \square

Theorem 2. MINIMUM SUBSTRING COVER is approximable within a factor of:

- $\binom{m+1}{2} - 1$, for general values of ℓ .
- $m - 1$, for $\ell = 2$.
- m , for $\ell = \infty$ and proper weight functions.

4 Linear Program Rounding

In [11], Hajiaghayi *et al.* considered the MINIMUM MULTICOLORED SUBGRAPH problem, which is a generalization of MINIMUM SUBSTRING COVER when the given factorization length ℓ is set to 2. In this section, we extend the linear program rounding algorithm given in [11] to apply for any constant values of ℓ . We obtain a $\mathcal{O}(\log n \cdot m^{(\ell-1)^2/\ell})$ -approximation algorithm for our problem, which outperforms the algorithm given in the previous section when $\ell < 4$. This algorithm can also be used for solving a generalization of the MINIMUM MULTICOLORED SUBGRAPH problem, namely

the MINIMUM MULTICOLORED HYPERGRAPH SUBGRAPH problem. Here ℓ corresponds to the size of the largest hyperedge, and m to the maximum number of hyperedges colored by any particular color.

Given a string s , an ℓ -factorization of s is an ordered multiset of substrings $f = (c_1, \dots, c_p)$ such that $s = c_1 \cdots c_p$ and $p \leq \ell$. Denote by $\mathcal{F}_\ell(s)$ the set of possible ℓ -factorizations of s , and let $\mathcal{F}_\ell(S)$ denote the set of all factorizations of strings in S , i.e. $\mathcal{F}_\ell(S) = \bigcup_{s \in S} \mathcal{F}_\ell(s)$. Now, for every substring $c \in \mathcal{C}(S)$, we designate a variable x_c which associated with c , and for every factorization $f \in \mathcal{F}_\ell(S)$, we designate a variable y_f which is associated with f . In these terms, MINIMUM SUBSTRING COVER can be formulated using the following integer linear program:

$$\begin{aligned}
\min \quad & \sum_{c \in \mathcal{C}(S)} w(c)x_c \\
\text{s.t.} \quad & \sum_{f \in \mathcal{F}_\ell(s)} y_f \geq 1 & \forall s \in S \\
& \sum_{c \in f \in \mathcal{F}_\ell(s)} y_f \leq x_c & \forall s \in S, \forall c \text{ substring of } s \\
& x_c, y_f \in \{0, 1\} & \forall c \in \mathcal{C}(S), \forall f \in \mathcal{F}_\ell(S)
\end{aligned} \tag{IP}$$

The variable x_c indicates whether the substring c is in the cover C and the variable y_f indicates whether C covers s by using the factorization f . The first type of constraints make sure that every string is factorized by some factorization. The second type of constraints make sure that if s is covered via the factorization f , then all substrings participating in this factorization are counted in the objective function. A linear programming relaxation of IP is obtained by replacing the integrality constraints by: (i) $x_c \geq 0$ for every $c \in \mathcal{C}(S)$, and (ii) $y_f \geq 0$ for every $f \in \mathcal{F}_\ell(S)$. Notice that the LP-relaxation is solvable in polynomial time since $\max_{s \in S} |\mathcal{F}_\ell(s)| = \mathcal{O}(m^{\ell-1})$, and ℓ is assumed to be constant. (It can also be solved for arbitrary values of ℓ by solving the dual).

Let $\mu > 1$ be a parameter to be determined later. Given an optimal fractional solution (x^*, y^*) to the LP-relaxation of IP, we construct an integral solution (x, y) for IP by picking every substring c with probability $p(c) = \min\{\mu \cdot x_c^*, 1\}$. That is, $x_c = 1$ with probability $p(c)$, and $x_c = 0$ with probability $1 - p(c)$. If there exists some $f \in \mathcal{F}_\ell(s)$ such that $x_c = 1$ for every $c \in f$ we set $y_f = 1$. We set $y_f = 0$ for any other $f \in \mathcal{F}_\ell(s)$. The resulting set of substrings is denoted by C , namely, $C = \{c : x_c = 1\}$.

The first step is to show that the expected total weight of our solution C is not much more than the total weight of the optimum cover of S . Let us denote the total weight of the optimal cover of S by OPT. We have:

Lemma 8. $\mathbf{E}[w(C)] \leq \mu \cdot \text{OPT}$.

Proof. $\mathbf{E}[w(C)] = \mathbf{E}[\sum_{c \in \mathcal{C}(S)} w(c)p(c)] \leq \sum_{c \in \mathcal{C}(S)} w(c)(\mu \cdot x_c^*) \leq \mu \cdot \text{OPT}$. □

The next step is to show that with a proper selection of μ , the probability that a string $s \in S$ is not covered by C becomes constant.

Lemma 9. For any string $s \in S$, if $\mu \geq |\mathcal{F}_\ell(s)|^{(\ell-1)/\ell}$ then $\Pr[C \text{ does not cover } s] \leq e^{-1}$.

Proof. Let $s \in S$ be any arbitrary string. We prove the lemma by suggesting an alternative method for covering s . For this we define the following three families of boolean random variables:

- $\{Z(f, c)\}_{c \in f \in \mathcal{F}(s)}$, where $\Pr[Z(f, c) = 1] = \min\{\mu \cdot y_f^*, 1\} = p(f)$.
- $\{X(c)\}_{c \in \mathcal{C}(\{s\})}$, where $X(c) = \bigvee_{c \in f \in \mathcal{F}(s)} Z(f, c)$.

– $\{Y(f)\}_{f \in \mathcal{F}_\ell(s)}$, where $Y(f) = \bigwedge_{c \in f} Z(f, c)$.

Note that all variables are independent within each family.

Our alternative method for covering s is done according to the variables $X(c)$. That is, we consider $C_s = \{c : X(c) = 1\}$ as our candidate set of substrings for covering s . We first show that the probability that C_s does not cover s is at least as high as the probability that C does not cover s . We do so by showing that $\Pr[X(c) = 1] \leq p(c)$ for every substring c of s . Indeed, if $p(c) = 1$ this is trivial. Also, if $p(f) = 1$ for some f with $c \in f$, then $p(c) = 1$. Otherwise, this follows by union bound and the feasibility of (x^*, y^*) with respect to IP:

$$\Pr[X(c) = 1] = \Pr\left[\bigvee_{c \in f \in \mathcal{F}(s)} Z(f, c) = 1\right] \leq \sum_{c \in f \in \mathcal{F}(s)} \Pr[Z(f, c) = 1] = \sum_{c \in f \in \mathcal{F}(s)} \mu \cdot y_f^* \leq \mu \cdot x_c^* = p(c).$$

We next show that C_s covers s with high probability. First, observe that if C_s does not cover s , then for any $f \in \mathcal{F}_\ell(s)$ there exists $c \in f$ such that $X(c) = 0$. From the definition of $X(c)$ it follows that $Z(f, c) = 0$ as well, and this means that $Y(f) = 0$ for every $f \in \mathcal{F}_\ell(s)$. Hence, $\Pr[C_s \text{ does not cover } s] \leq \Pr[\forall f \in \mathcal{F}_\ell(s) : Y(f) = 0]$. We have,

$$\Pr[\forall f \in \mathcal{F}_\ell(s) : Y(f) = 0] = \prod_{f \in \mathcal{F}_\ell(s)} \Pr[Y(f) = 0] \leq \prod_{f \in \mathcal{F}_\ell(s)} (1 - p(f)^\ell) \leq \prod_{f \in \mathcal{F}_\ell(s)} e^{-p(f)^\ell} = e^{-\sum_{f \in \mathcal{F}_\ell(s)} p(f)^\ell},$$

where the second inequality is due to the fact that $1 - x \leq e^{-x}$ for $x \in [0, 1]$. Now observe that, for any $f \in \mathcal{F}_\ell(s)$, if $p(f) = 1$ then $Y(f) = 1$. Hence, since $p(f) = \mu \cdot y_f^* < 1$ for all $f \in \mathcal{F}_\ell(s)$, and since y^* is a feasible solution of IP, we have $\sum_{f \in \mathcal{F}_\ell(s)} p(f) = \mu \cdot \sum_{f \in \mathcal{F}_\ell(s)} y_f^* \geq \mu$ for all $f \in \mathcal{F}_\ell(s)$. Due to this, and the convexity of the function $f(x) = x^\ell$ for $x \in [0, 1]$, we get that

$$\sum_{f \in \mathcal{F}_\ell(s)} p(f)^\ell \geq |\mathcal{F}_\ell(s)| \left(\frac{\sum_{f \in \mathcal{F}_\ell(s)} p(f)}{|\mathcal{F}_\ell(s)|} \right)^\ell \geq \frac{\mu^\ell}{|\mathcal{F}_\ell(s)|^{\ell-1}} \geq \frac{|\mathcal{F}_\ell(s)|^{\ell-1}}{|\mathcal{F}_\ell(s)|^{\ell-1}} = 1,$$

and so $\Pr[C_s \text{ does not cover } s] \leq \Pr[\forall f \in \mathcal{F}_\ell(s), Y(f) = 0] \leq e^{-1}$. \square

The previous lemma implies that by setting $\mu = \max_{s \in S} |\mathcal{F}_\ell(s)|^{(\ell-1)/\ell} = \mathcal{O}(m^{(\ell-1)^2/\ell})$, we cover any string with constant probability. However, we would like to cover all strings in S . To achieve this goal we can execute the rounding algorithm $t = \log n + 2$ times independently in order to obtain t candidate solutions C_1, \dots, C_t , and output $C = \cup_{i=1}^t C_i$. Clearly, $\mathbf{E}[w(C)] \leq (\log n + 2)\mu \cdot \text{OPT}$. Hence, by Markov's inequality, $\Pr[w(C) \geq 2(\log n + 2)\mu \cdot \text{OPT}] \leq 1/2$. Furthermore, $\Pr[\exists s \text{ not covered by } C] \leq n \cdot e^{-(\log n + 2)} = e^{-2}$. Hence, by using some more amplification, we obtain the main result of this section:

Theorem 3. *With high probability, MINIMUM SUBSTRING COVER is approximable within a factor of $\mathcal{O}(\log n \cdot m^{(\ell-1)^2/\ell})$.*

References

1. P. Alimonti and V. Kann. Hardness of approximating problems on cubic graphs. In *Proceedings of the 3rd Italian Conference on Algorithms and Complexity (CIAC)*, pages 288–298, 1997.
2. R. Bar-Yehuda. One for the price of two: A unified approach for approximating covering problems. *Algorithmica*, 27(2):131–144, 2000.

3. R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203, 1981.
4. R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.
5. H.L. Bodlaender, R.G. Downey, M.R. Fellows, M.T. Hallett, and H.T. Wareham. Parameterized complexity analysis in computational biology. *Computer Applications in the Biosciences*, 11(1):49–57, 1995.
6. C. Choffrut and J. Karhumäki. *Combinatorics of Words*, in G. Rozenberg and A. Salomaa (eds), *Handbook of Formal Languages*. Springer-Verlag, 1997.
7. V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.
8. I. Dinur, V. Guruswami, S. Khot, and O. Regev. A new multilayered PCP and the hardness of hypergraph vertex cover. *SIAM Journal on Computing*, 34(5):1129–1146, 2005.
9. R.L. Dorit and W. Gilbert. The limited universe of exons. *Current Opinions in Structural Biology*, 1:973–977, 1991.
10. A. Ehrenfeucht and G. Rozenberg. Elementary homomorphisms and a solution of the D0L sequence equivalence problem. *Theoretical Computer Science*, 7:169–183, 1978.
11. M.T. Hajiaghayi, K. Jain, L.C. Lau, I.I. Mandoiu, A. Russell, and V.V. Vazirani. Minimum multicolored subgraph problem in multiplex PCR primer set selection and population haplotyping. In *Proceedings of the 6th International Conference on Computational Science (ICCS)*, pages 758–766, 2006.
12. D.S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.
13. D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.
14. L. Lovász. On the ratio of optimal integral and fractional solutions. *Discrete Mathematics*, 13:383–390, 1974.
15. J. Néraud. Elementariness of a finite set of words is co-NP-complete. *Theoretical Informatics and Applications*, 24(5):459–470, 1990.
16. C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and Systems Sciences*, 43:425–440, 1991.
17. L. Patthy. Exons - original building blocks of proteins? *BioEssays*, 13(4):187–192, 1991.
18. R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th ACM Symposium on the Theory Of Computing (STOC)*, pages 475–484, 1997.
19. G. Rozenberg and A. Salomaa. *The Mathematical Theory of L Systems*. Academic Press, New York, 1980.