



# The Minimum Substring Cover problem<sup>☆</sup>

Danny Hermelin<sup>a,1</sup>, Dror Rawitz<sup>b,\*,1</sup>, Romeo Rizzi<sup>c,2</sup>, Stéphane Vialette<sup>d,2</sup>

<sup>a</sup> Department of Computer Science, University of Haifa, Haifa 31905, Israel

<sup>b</sup> School of Electrical Engineering, Tel-Aviv University, Tel-Aviv 69978, Israel

<sup>c</sup> Dipartimento di Matematica ed Informatica (DIMI), Università di Udine, I-33100 Udine, Italy

<sup>d</sup> Laboratoire de Recherche en Informatique (LRI), Université Paris-Sud, 91405 Orsay, France

## ARTICLE INFO

### Article history:

Received 21 April 2007

Revised 14 May 2008

Available online 26 June 2008

### Keywords:

Approximation algorithms

Dictionary Generation

Local-ratio

Randomized rounding

Substring Cover

## 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 several hardness of approximation results for the problem, and in the main part of the paper, we focus on devising approximation algorithms using two generic paradigms—the local-ratio technique and linear programming rounding.

© 2008 Published by Elsevier Inc.

## 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 [15] and Lovász [16] 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 [13] was also designed for *MINIMUM SET COVER*, while Bar-Yehuda and Even gave the first combinatorial 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 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  $\mathcal{C}(S)$

<sup>☆</sup> An extended abstract appeared in the 5th Workshop on Approximation and Online Algorithms (WAOA), 2007.

\* Corresponding author.

E-mail addresses: [danny@cri.haifa.ac.il](mailto:danny@cri.haifa.ac.il) (D. Hermelin), [rawitz@eng.tau.ac.il](mailto:rawitz@eng.tau.ac.il) (D. Rawitz), [Romeo.Rizzi@dimi.uniud.it](mailto:Romeo.Rizzi@dimi.uniud.it) (R. Rizzi), [viallette@lri.fr](mailto:viallette@lri.fr) (S. Vialette).

<sup>1</sup> Partially supported by the Caesarea Rothschild Institute (CRI).

<sup>2</sup> Supported by the Italian-French PAI Galileo Project 08484VH.

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  $\ell$  strings in  $C$ , we say that  $C$  is an  $\ell$ -*cover* of  $S$ . Given a weight function  $w : \mathcal{C}(S) \rightarrow \mathcal{Q}^+$ , we are interested in computing an  $\ell$ -cover of  $S$  with minimum possible weight:

MINIMUM SUBSTRING COVER:

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

*Solution:* An  $\ell$ -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  $S = \{ 'a', 'aab', 'aba' \}$ . Then  $\mathcal{C}(S) = \{ 'a', 'b', 'aa', 'ab', 'ba', 'aab', 'aba' \}$ , and  $C_1 = \{ 'a', 'b' \}$  and  $C_2 = \{ 'a', 'ab' \}$  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  $m = \max\{|s| : s \in S\}$ . Notice that  $|\mathcal{C}(S)| = \mathcal{O}(\sum_{s \in S} |s|^2) = \mathcal{O}(nm^2)$ .

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  $\infty$ -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  $w : \mathcal{C}(S) \rightarrow \mathcal{Q}^+$  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 [9,19].

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]). Néraud [17] 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. Néraud describes a direct application of this notion to the famous DOL-sequence equivalence problem (cf. [21]) 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.

Apart from the work of Bodlaender et al. [5] and Néraud [17], there has also been some recent work on problems closely related to MINIMUM SUBSTRING COVER, especially for the case of  $\ell = 2$ . The MINIMUM SET COVER WITH PAIRS problem introduced by Hassin and Segev in [12], is a variant of MINIMUM SET COVER where base elements are now covered by pairs of sets, and the goal is to cover all base elements using a minimum weight collection of sets. Hassin and Segev gave an  $\mathcal{O}(\sqrt{n} \lg n)$ -approximation algorithm for the unweighted version of this problem, where  $n$  is the number of elements, along with a few other algorithms for special cases of this problem. Another closely related problem is the HAPLOTYPE INFERENCE BY MAXIMUM PARSIMONY, an important problem in computational biology. Huang et al. [14] gave an algorithm for this problem, which translates to an  $\mathcal{O}(\lg n \cdot m^2)$  algorithm for MINIMUM SUBSTRING COVER with  $\ell = 2$ . Hajiaghayi et al. [11] introduced the MINIMUM MULTICOLORED SUBGRAPH problem within the same context, and gave a randomized algorithm which in our

terms obtains a performance ratio of  $\mathcal{O}(\sqrt{m \cdot \lg n})$  with constant probability. We discuss this algorithm and how to extend it to MINIMUM SUBSTRING COVER with general values of  $\ell$  in Section 4.

The rest of this paper is organized as follows.

- 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  $\lfloor m/2 \rfloor - 1 - \varepsilon$  for 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  $\mathcal{O}(m^{(\ell-1)^2/\ell} \cdot \lg^{1/\ell} n)$  with high probability. This algorithm is an extension of an algorithm due to Hajiaghayi et al. [11].

## 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  $w : \mathcal{C}(S) \rightarrow \mathcal{Q}^+$  is either the unitary or the length-weighted function.

To prove our approximation lower bounds, we present an L-reduction [18] 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  $H = (V(H), E(H))$ ,  $w_H : V(H) \rightarrow \mathcal{Q}^+$ , and the goal is to find a minimum weight vertex cover of  $H$ . That is, a subset of vertices  $U \subseteq V(H)$  of minimum weight, such that  $U \cap e \neq \emptyset$  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  $c$  [20], 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  $(S, w, \ell)$  for MINIMUM SUBSTRING COVER as follows. Let  $e_{\max}$  denote an edge of maximal size in  $H$ . The set of strings  $S$  is defined over an alphabet  $\Sigma$  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 ‘ $v v'$ ’ 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  $w : \mathcal{C}(S) \rightarrow \mathcal{Q}^+$  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  $k$  if and only if  $S$  has an  $\ell$ -cover with total weight  $k$ .

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

Conversely, suppose  $S$  has an  $\ell$ -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  $\ell$ -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  $U \subseteq V(H)$  be the vertices in  $H$  corresponding to the encodings in  $C_2$ . Then  $w_H(U) = w(C_2) = k$ . 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  $U \cap e \neq \emptyset$  for all  $e \in E(H)$ .  $\square$

The lemma above implies that any  $\alpha$ -approximation algorithm for MINIMUM SUBSTRING COVER would give an  $\alpha$ -approximation algorithm for MINIMUM HYPERGRAPH VERTEX COVER. Hence, due to the results of [20] 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  $E(G)$  [1,18]. 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  $k$  if and only if  $S$  has a 3-cover of size  $2|V(G)| + k$ .*

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

Conversely, suppose  $S$  has a 3-cover  $C$ . Since  $\Sigma \subseteq C$ ,  $C$  must include every letter in  $\Sigma$ . 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  $U \subseteq V(G)$  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  $\Sigma$ . 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  $k$  if and only if  $S$  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  $c > 1$ , when  $m$  and  $\ell$  are constant, and  $w$  is either the unitary or the length-weighted function.

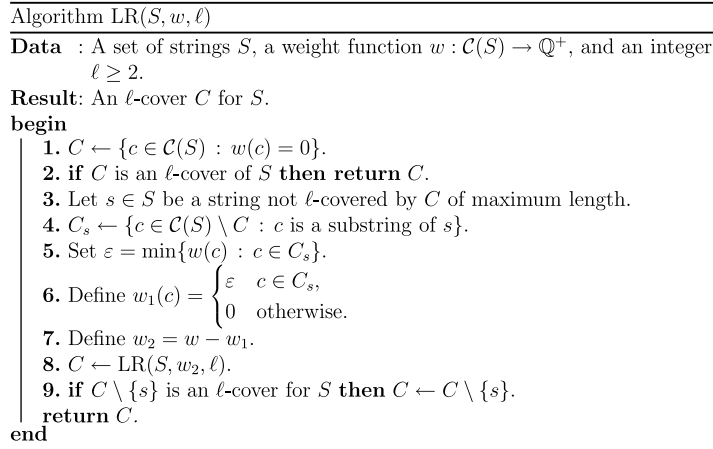
### 3. Local-Ratio Algorithms

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 approximation 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  $\ell$ -cover for  $S$  for general values of  $\ell$ . 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.

#### 3.1. 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  $C(S)$ . If  $C$  is an  $\alpha$ -approximate, both with respect to  $w_1$  and with respect to  $w_2$ , then  $C$  is also  $\alpha$ -approximate with respect to  $w_1 + w_2$ .*



**Fig. 1.** A local-ratio approximation framework.

A local-ratio  $\alpha$ -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  $\alpha$ -approximate with respect to  $w_1$ , and (2) there exists an element  $e$  in our instance 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  $\alpha$ -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.

### 3.2. Approximation algorithm

Fig. 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  $\left(\binom{m+1}{2} - 1\right)$ -approximate  $\ell$ -covers for general values of  $\ell$ . 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.

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  $\left(\binom{m+1}{2} - 1\right)$ -approximate with respect to  $w_1$ , and also with respect to  $w_2$ , and therefore due to the Local-Ratio Lemma, are also  $\left(\binom{m+1}{2} - 1\right)$ -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  $\ell$ -cover of  $S$ . In the following lemma we show that this cover is  $\left(\binom{m+1}{2} - 1\right)$ -approximate.

**Lemma 5.** Algorithm LR computes an  $\left(\binom{m+1}{2} - 1\right)$ -approximate  $\ell$ -cover of  $S$ .

**Proof.** To prove that the cover  $C$  returned by algorithm LR is  $\left(\binom{m+1}{2} - 1\right)$ -approximate, we apply induction on the number of recursive calls of the algorithm, and show that at any recursive call,  $C$  is  $\left(\binom{m+1}{2} - 1\right)$ -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  $\left(\binom{m+1}{2} - 1\right)$ -approximate. For the inductive step, consider any recursive call other than the basis, and assume that the cover  $C$  returned at Line 8 is  $\left(\binom{m+1}{2} - 1\right)$ -approximate with respect to  $w_2$ . Note that  $C$  also remains  $\left(\binom{m+1}{2} - 1\right)$ -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,  $\sum_{c \in C(S)} w_1(c) \leq \binom{m+1}{2} \varepsilon$ . 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  $C(S)$ . 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,  $\sum_{c \in C} w_1(c) \leq \left(\binom{m+1}{2} - 1\right) \varepsilon$ . Furthermore, by our selection of  $\varepsilon$ , any cover for  $S$  has weight at least  $\varepsilon$  with respect to  $w_1$ . It follows that, after Line 9,  $C$  is  $\left(\binom{m+1}{2} - 1\right)$ -approximate with respect to  $w_1$  as well as with respect to  $w_2$ . According to the Local-Ratio Lemma, the cover returned is  $\left(\binom{m+1}{2} - 1\right)$ -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  $C(S)$  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:

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

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  $LR_2$ .

It is clear that algorithm  $LR_2$  can be implemented to run in polynomial time. Furthermore, the analysis of the performance ratio of algorithm  $LR_2$  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  $w : C(S) \rightarrow \mathbb{Q}^+$ , we say that  $w$  is *proper* if for any  $c, c_1 \in C(S)$ ,  $w(c_1) \leq w(c)$  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  $LR_\infty$ . It is obtained by replacing Line 9 in algorithm LR with the following line:

$$\text{while } \exists c, c_1 \in C \text{ with } c = c_1 c_2 \text{ or } c = c_2 c_1 \text{ do } C \leftarrow (C \setminus \{c\}) \cup \{c_2\}.$$

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  $\ell$ -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  $LR_\infty$  computes an  $m$ -approximate cover of  $S$  assuming the given weight function  $w : C(S) \rightarrow \mathbb{Q}^+$  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  $LR_\infty$  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  $\varepsilon \leq \sum_{c \in C} w_1(c) \leq m \cdot \varepsilon$ .  $\square$

We summarize the results obtained in this section in the following theorem:

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

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

#### 4. Linear programming 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  $\ell$  is set to 2. In this section, we extend the linear programming rounding algorithm given in [11] to apply for any constant value of  $\ell$ .  $\mathcal{O}(m^{(\ell-1)^2/\ell} \cdot \lg^{1/\ell} n)$ -approximation algorithm for our problem, which outperforms the algorithm given in the previous section when  $\ell < 4$ . (Note that when  $\ell \geq 4$  the approximation guarantee is  $\omega(m^{9/4})$ .)

In the MINIMUM MULTICOLORED SUBGRAPH problem the input consists of an undirected graph  $G$  with non-negative vertex weights and a color function that assigns to each edge one or more of  $k$  given colors. The goal in this problem is to find a minimum weight set of vertices inducing edges of all  $k$  colors. MINIMUM SUBSTRING COVER in the case where  $\ell = 2$  can be reduced to MINIMUM MULTICOLORED SUBGRAPH as follows. Given a set of strings  $S$  we construct a graph  $G$  whose vertices correspond to prefixes and suffixes of strings in  $S$ . Each string  $s \in S$  corresponds to a color  $\chi_s$ , and edges that correspond to a string  $s$  are colored by  $\chi_s$ . Hajiaghayi et al. [11] gave an algorithm whose approximation ratio is  $\mathcal{O}(\sqrt{M} \cdot \log k)$ , where  $M$  is the maximum number of edges colored by any particular color. In MINIMUM SUBSTRING COVER terms this amounts to  $\mathcal{O}(\sqrt{m \cdot \lg n})$ .

The algorithm that we present in this section can also be used for solving the MINIMUM MULTICOLORED SUBGRAPH problem in hypergraphs. Here  $\ell$  corresponds to the size of the largest hyperedge, and the maximum number of hyperedges colored by any particular color  $M$  replaces  $\mathcal{O}(m^{\ell-1})$ . Hence, the approximation ratio is  $\mathcal{O}((M^{\ell-1} \cdot \lg n)^{1/\ell})$ .

##### 4.1. Linear programming formulation

Given a string  $s$ , an  $\ell$ -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  $\ell$ -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:

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

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  $\ell$  is assumed to be constant.

##### 4.2. Approximation algorithm

Let  $\mu = \mu(n, m, \ell) > 1$  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 independently 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  $1 - p(c)$ . For every  $s \in S$ , 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.**  $\mathbf{E}[w(C)] = \mathbf{E}\left[\sum_{c \in C(S)} w(c)x_c\right] = \sum_{c \in C(S)} w(c)p(c) \leq \sum_{c \in C(S)} w(c)(\mu \cdot x_c^*) \leq \mu \cdot \text{OPT}$ .  $\square$

The next step is to show that with a proper selection of  $\mu$ , the probability that a string  $s \in S$  is not covered by  $C$  is  $o(1)$ .

**Lemma 9.** For any string  $s \in S$ , if

$$\mu \geq (\ln n + 1)^{1/\ell} \cdot |\mathcal{F}_\ell(s)|^{(\ell-1)/\ell}$$

then

$$\Pr[C \text{ does not cover } s] \leq (e \cdot n)^{-1}.$$

**Proof.** Let  $s \in S$  be any arbitrary string. We prove the lemma by suggesting an alternative method for covering  $s$ . Namely, we construct a random set of substrings using  $y^*$  instead of  $x^*$ . For this we define the following three families of boolean random variables:

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

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

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,  $\Pr[C_s \text{ does not cover } s] \leq \Pr[\forall f \in \mathcal{F}_\ell(s) : Y(f) = 0]$ .

Now observe that, for any  $f \in \mathcal{F}_\ell(s)$ , if  $p(f) = 1$  then  $Y(f) = 1$ . Hence, for the rest of the proof we assume that  $p(f) < 1$ . We have,

$$\begin{aligned} \Pr[\forall f \in \mathcal{F}_\ell(s) : Y(f) = 0] &= \prod_{f \in \mathcal{F}_\ell(s)} \Pr[Y(f) = 0] \\ &= \prod_{f \in \mathcal{F}_\ell(s)} \Pr\left[\bigwedge_{c \in f} Z(f,c) = 0\right] \\ &= \prod_{f \in \mathcal{F}_\ell(s)} \left(1 - \Pr\left[\bigvee_{c \in f} Z(f,c) = 1\right]\right) \\ &\leq \prod_{f \in \mathcal{F}_\ell(s)} (1 - p(f)^\ell) \\ &\leq \prod_{f \in \mathcal{F}_\ell(s)} e^{-p(f)^\ell} \\ &= e^{-\sum_{f \in \mathcal{F}_\ell(s)} p(f)^\ell}, \end{aligned}$$

where the second inequality is due to the fact that  $1 - x \leq e^{-x}$ . Since  $p(f) = \mu \cdot y_f^* < 1$  for all  $f \in \mathcal{F}_\ell(s)$ , and since  $(x^*, y^*)$  is a feasible solution of the LP relaxation 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 the convexity of the function  $f(x) = x^\ell$  for  $x \in [0,1]$ , we get that

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

and so

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

and we are done.  $\square$

The previous lemma states that by setting

$$\mu = (\ln n + 1)^{1/\ell} \cdot \max_{s \in S} |\mathcal{F}_\ell(s)|^{(\ell-1)/\ell} = \mathcal{O}(m^{(\ell-1)^2/\ell} \cdot \lg^{1/\ell} n)$$

we cover any string  $s \in S$  with probability at least  $\frac{1}{e \cdot n}$ . Due to union bound it follows that

$$\Pr[\exists s \text{ not covered by } C] \leq \frac{n}{e \cdot n} = e^{-1}.$$

We now bound the weight of  $C$  in case it covers the strings in  $S$ . Notice that Lemma 8 bounds the expected weight of  $C$ , where the expectancy is taken over *all* subsets of  $C$  and not only over those which are feasible solutions.

**Lemma 10.**  $\mathbf{E}[w(C)|S \text{ is covered by } C] \leq \mu \cdot \text{OPT}$

**Proof**

$$\begin{aligned} \mathbf{E}[w(C)] &= \Pr[S \text{ is covered by } C] \cdot \mathbf{E}[w(C)|S \text{ is covered by } C] \\ &\quad + \Pr[\exists s \text{ not covered by } C] \cdot \mathbf{E}[w(C)|\exists s \text{ not covered by } C] \\ &\geq (1 - 1/e) \cdot \mathbf{E}[w(C)|S \text{ is covered by } C] \end{aligned}$$

which means that

$$\mathbf{E}[w(C)|S \text{ is covered by } C] = \frac{\mathbf{E}[w(C)]}{1 - 1/e} \leq \frac{\mu \cdot \text{OPT}}{1 - 1/e},$$

where the last inequality is due to Lemma 8.

Hence, the rounding algorithm computes an  $\mathcal{O}(m^{(\ell-1)^2/\ell} \cdot \lg^{1/\ell} n)$ -approximate  $\ell$ -cover with probability  $1 - 1/e$ . To obtain the same approximation ratio with high probability. We run the rounding algorithm  $\lceil \ln n \rceil$  times and take the best  $\ell$ -cover among up to  $\lceil \ln n \rceil$  computed  $\ell$ -covers. The probability of failure is at most  $(e^{-1})^{\lceil \ln n \rceil} \approx 1/n$ , and the expected weight of the computed  $\ell$ -cover remains  $\mathcal{O}(m^{(\ell-1)^2/\ell} \cdot \lg^{1/\ell} n) \cdot \text{OPT}$ .

**Theorem 3.** *There exists a randomized algorithm for MINIMUM SUBSTRING COVER that computes an  $\mathcal{O}(m^{(\ell-1)^2/\ell} \cdot \lg^{1/\ell} n)$ -approximation with high probability.*

### Acknowledgment

We thank Danny Segev for helpful remarks.

### References

- [1] P. Alimonti, V. Kann, Hardness of approximating problems on cubic graphs, in: Proceedings of the 3rd Italian Conference on Algorithms and Complexity (CIAC), 1997.
- [2] R. Bar-Yehuda, One for the price of two: a unified approach for approximating covering problems, *Algorithmica*, 27 (2) (2000) 131–144.
- [3] R. Bar-Yehuda, S. Even, A linear time approximation algorithm for the weighted vertex cover problem, *Journal of Algorithms* 2 (1981) 198–203.
- [4] R. Bar-Yehuda, S. Even, A local-ratio theorem for approximating the weighted vertex cover problem, *Annals of Discrete Mathematics* 25 (1985) 27–46.
- [5] H. Bodlaender, R. Downey, M. Fellows, M. Hallett, H. Wareham, Parameterized complexity analysis in computational biology, *Computer Applications in the Biosciences* 11 (1) (1995) 49–57.
- [6] C. Choffrut, J. Karhumäki, Combinatorics of words, in: G. Rozenberg, 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) (1979) 233–235.
- [8] I. Dinur, V. Guruswami, S. Khot, O. Regev, A new multilayered PCP and the hardness of hypergraph vertex cover, *SIAM Journal on Computing* 34 (5) (2005) 1129–1146.

- [9] R. Dorit, W. Gilbert, The limited universe of exons, *Current Opinions in Structural Biology* 1 (1991) 973–977.
- [10] A. Ehrenfeucht, G. Rozenberg, Elementary homomorphisms and a solution of the D0L sequence equivalence problem, *Theoretical Computer Science* 7 (1978) 169–183.
- [11] M. Hajiaghayi, K. Jain, L. Lau, I. Mandoiu, A. Russell, 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)*, 2006.
- [12] R. Hassin, D. Segev, The set cover with pairs problem, in: *Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2005.
- [13] D. Hochbaum, Approximation algorithms for the set covering and vertex cover problems, *SIAM Journal on Computing* 11 (3) (1982) 555–556.
- [14] Y.-T. Huang, K.-M. Chao, T. Chen, An approximation algorithm for haplotype inference by maximum parsimony, in: *Proceedings of the 20th ACM Symposium on Applied Computing (SAC)*, 2005.
- [15] D. Johnson, Approximation algorithms for combinatorial problems, *Journal of Computer and System Sciences* 9 (1974) 256–278.
- [16] L. Lovász, On the ratio of optimal integral and fractional solutions, *Discrete Mathematics* 13 (1974) 383–390.
- [17] J. Néraud, Elementariness of a finite set of words is co-NP-complete, *Theoretical Informatics and Applications* 24 (5) (1990) 459–470.
- [18] C. Papadimitriou, M. Yannakakis, Optimization, approximation, and complexity classes, *Journal of Computer and Systems Sciences* 43 (1991) 425–440.
- [19] L. Patthy, Exons—original building blocks of proteins?, *BioEssays* 13 (4) (1991) 187–192.
- [20] R. Raz, 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)*, 1997.
- [21] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, New York, 1980.