New Complexity Results for the k-Covers Problem

Costas S. Iliopoulos[1⋆], Manal Mohamed[1⋆⋆], and W.F. Smyth[2⋆⋆⋆]

[1] Department of Computer Science, King's College London
London WC2R 2LS, England
csi@dcs.kcl.ac.uk
www.dcs.kcl.ac.uk/staff/csi

[2] Algorithms Research Group, Department of Computing & Software
McMaster University, Hamilton, Ontario, Canada L8S 4K1

Department of Computing, Curtin University, Perth WA 6845, Australia
smyth@mcmaster.ca
www.dcss.mcmaster.ca/~ bill/cv.shtml

Abstract. The k-covers problem (kCP) asks us to compute a minimum cardinality set of strings of given length $k > 1$ that covers a given string. It was shown in a recent paper, by reduction to 3-SAT, that the k-covers problem is NP-complete. In this paper we introduce a new problem, that we call the Relaxed Vertex Cover Problem (RVCP), which we show is a special case of Set Cover (SCP). We show further that kCP is equivalent to RVCP restricted to certain classes $G_{\boldsymbol{x},k}$ of graphs that represent all strings $\boldsymbol{x}$. We discuss approximate solutions of kCP, and we state a number of conjectures and open problems related to kCP and $G_{\boldsymbol{x},k}$.

Keywords: String, word, cover, regularity, complexity, NP-complete.

1 Introduction

The computation of various kinds of "regularities" in given strings $\boldsymbol{x} = \boldsymbol{x}[1..n]$ has been of interest for a quarter-century, signalled by the publication in the early 1980s of several $O(n \log n)$-time algorithms for computing all ***repetitions*** (adjacent identical substrings) [C81,AP83,ML84], work that has more recently been refined to $O(n)$-time algorithms [M89,KK00]. In response to applications arising in data compression and molecular biology, the computation of repetitions was generalized to computation of ***repeats*** (adjacency condition dropped), for which also $O(n)$-time algorithms have been found [BLPS00,FST03]; then still further to computation of ***approximate repeats*** [S98].

⋆ Supported in part by a Marie Curie fellowship, Wellcome & Royal Society grants.
⋆⋆ Supported by an EPSERC studentship.
⋆⋆⋆ Supported in part by a grant from the Natural Sciences & Engineering Research Council of Canada.

In [AFI91] the idea of a ***quasiperiod*** or ***cover*** was introduced; that is, a proper substring $\boldsymbol{u}$ of the given string $\boldsymbol{x}$ such that every position of $\boldsymbol{x}$ is contained in an occurrence of $\boldsymbol{u}$. Several algorithms to compute covers of $\boldsymbol{x}$ were published in the 1990s, culminating in an algorithm [LS02] that in $O(n)$ time computes a ***cover array*** specifying all the covers (quasiperiods) of every prefix of $\boldsymbol{x}$; this algorithm thus directly generalizes the border array ("failure function") algorithm [AHU74] that specifies all the borders, hence all the periods, of every prefix of $\boldsymbol{x}$.

In [IS98] a further extension, the ***k-covers problem***, was introduced: compute a minimum set $U_\nu = \{\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_\nu}\}$ of strings of given length $k > 1$ such that every position of $\boldsymbol{x}$ is contained in an occurrence of some element of U_ν. A polynomial-time algorithm was given for this problem, later discovered to be incorrect [Y00]; just recently the problem itself has been shown to be NP-complete, based on a reduction to 3-SAT [CIMSY03]. In this latter paper, two $O(n \log n)$ algorithms were described that yielded an approximation to a minimum k-cover of $\boldsymbol{x}$; it was conjectured that these algorithms would yield a k-cover of cardinality at most $\log n$ times the minimum.

In Section 2 of this paper we show that the k-covers problem is reducible to a subproblem of the Set Cover Problem (SCP). Thus the existence of an approximation algorithm that achieves at least a logarithmic multiple of the minimum cover is assured [J74,L75]. In fact, the reduction of k-covers is to a new problem, a special case of SCP that we call the Relaxed Vertex Cover Problem (RVCP). Since it is well-known [GJ79,S82] that the Vertex Cover Problem (VCP) actually has an approximation algorithm that achieves at most twice the minimum cover, the new reduction of k-covers raises the possibility that in fact k-covers also can also be approximated within a factor of two.

In Section 3 we discuss conjectures and open problems derived from the complexity analysis of the k-covers problem, both here and in [CIMSY03].

2 The Relaxed Vertex Cover Problem

Here we consider the decision form of the k-covers problem: given a string $\boldsymbol{x}$ and integers $k > 1$ and ν, decide whether there exists a k-cover of $\boldsymbol{x}$ of cardinality ν. We call this problem kCP and we show that it is a special case of SCP.

A ***vertex cover*** of an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ such that if $(u, v) \in E$, then either $u \in V'$ or $v \in V'$. The ***vertex cover problem*** (VCP) asks us to determine, for given graph G and integer ν, whether there exists a vertex cover V' of G such that $|V'| = \nu$. VCP is well-known to be NP-complete [GJ79], but several $O(|V| + |E|)$-time greedy algorithms have been described that compute a vertex cover of G whose cardinality is at most twice that of the minimum-cardinality vertex cover of G. The first such algorithm discovered, due to Gavril ([GJ79], p. 134), is a straightforward application of maximal matching; another [S82] is based on depth-first search.

It is convenient here to define a relaxed version of VCP. We say that a ***vertex semi-cover*** of $G = (V, E)$ is a subset $V' \subseteq V$ such that if $(u, v) \in E$, then one of the following conditions holds:

(C1) $u \in V'$;
(C2) $v \in V'$;
(C3) there exist $w_u, w_v \in V'$ such that $(w_u, u) \in E$ **and** $(w_v, v) \in E$.

The ***relaxed VCP*** (RVCP) asks, for given G and ν, whether there exists a vertex semi-cover of G such that $|V'| = \nu$. Of course every vertex cover of G is also a vertex semi-cover of G; thus, if ν^*_{VC} (respectively, ν^*_{RVC}) is the minimum cardinality of a vertex cover (respectively, semi-cover) V' of G, then

$$\nu^*_{RVC} \leq \nu^*_{VC}. \tag{1}$$

Since as just remarked an efficient approximation algorithm can be found for VCP that computes

$$\nu^{EST}_{VC} \in \nu^*_{VC}..2\nu^*_{VC}, \tag{2}$$

it follows from (1) that the same algorithm will also provide an upper bound for RVCP:

$$\nu^*_{RVC} \leq \nu^{EST}_{VC} \leq 2\nu^*_{VC}. \tag{3}$$

Suppose now that a string $\boldsymbol{x} = \boldsymbol{x}[1..n]$ and an integer k are given. We initialize a graph $G_{\boldsymbol{x},k} = (V, E)$, where $V = \{1, 2, \ldots, n\}$ and $E = \emptyset$. At every position $i = 1, 2, \ldots, n-k+1$ of $\boldsymbol{x}$, determine the leftmost position $i' \leq i$ in $\boldsymbol{x}$ at which the substring $\boldsymbol{u} = \boldsymbol{x}[i..i+k-1]$ occurs: if $i' = i$, compute

$$E \leftarrow E \cup \big\{(i', i+1), (i', i+2), \ldots, (i', i+k-1)\big\};$$

while if $i' < i$, compute

$$E \leftarrow E \cup \big\{(i', i), (i', i+1), \ldots, (i', i+k-1)\big\}.$$

Thus the only edges in $|E|$ are those that connect the leftmost position of each distinct k-string $\boldsymbol{u}$ with every other position in $\boldsymbol{x}$ covered by $\boldsymbol{u}$; hence

$$(k-1)(n-k+1) \leq |E| \leq k(n-k+1) - 1.$$

For example, given $\boldsymbol{x} = abaab$ and $k = 2$, the corresponding graph $G_{\boldsymbol{x},k}$ will be given by

$$V = \{1, 2, 3, 4, 5\}, \quad E = \big\{(1, 2), (1, 4), (1, 5), (2, 3), (3, 4)\big\}.$$

Thus the position of the leftmost occurrence of every distinct k-string $\boldsymbol{u}$ of $\boldsymbol{x}$ will be adjacent to every other position covered by $\boldsymbol{u}$. If the alphabet of $\boldsymbol{x}$ is ordered, an algorithm to compute $G_{\boldsymbol{x},k}$ from $\boldsymbol{x}$ can be implemented in $O(kn \log n)$ time using a straightforward approach, somewhat faster using a suffix tree to sort the k-strings. Let us designate this algorithm $A(\boldsymbol{x}, k)$.

Consider a set $U_\nu = \{\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_\nu}\}$ of distinct substrings of $\boldsymbol{x}$, each of length k, and let j_i denote the leftmost position in $\boldsymbol{x}$ at which $\boldsymbol{u_i}$ occurs, $i = 1, 2, \ldots, \nu$. Let $J_\nu = \{j_1, j_2, \ldots, j_\nu\}$. Now observe that if in fact U_ν is a k-cover of $\boldsymbol{x}$, a vertex semi-cover of $G_{\boldsymbol{x},k}$ is given by $V' = J_\nu$: for every $(u, v) \in E$, if neither (C1) nor (C2) is true, then by the construction of E, (C3) must be true. Conversely, if $V' = \{j_1, j_2, \ldots, j_\nu\}$ is a vertex semi-cover of a graph $G_{\boldsymbol{x},k}$ formed using $A(\boldsymbol{x}, k)$ for some string $\boldsymbol{x}$ and some $k \geq 2$, then

$$
\begin{aligned}
U'_\nu &= \big\{\boldsymbol{x}[j_1..j_1+k-1], \boldsymbol{x}[j_2..j_2+k-1], \ldots, \boldsymbol{x}[j_\nu..j_\nu+k-1]\big\} \\
&= \{\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_\nu}\}
\end{aligned}
$$

must be a k-cover of $\boldsymbol{x}$. Thus:

Theorem 1. *Problem kCP is equivalent to RVCP restricted to graphs $G_{\boldsymbol{x},k}$ formed according to Algorithm $A(\boldsymbol{x}, k)$.* $\square$

Since as proved in [CIMSY03], kCP is NP-complete over strings $\boldsymbol{x}$ for every fixed choice of $k \geq 2$, we have immediately:

Theorem 2. *RVCP restricted to graphs computed using Algorithm $A(\boldsymbol{x}, k)$, $k \geq 2$, is NP-complete.* $\square$

It is not hard to show that RVCP, like VCP, is a special case of SCP:

Theorem 3. *Every instance of RVCP is equivalent to an instance of SCP.*

Proof. Suppose we are given a graph $G = (V, E)$ together with a subset V' of V. We construct a set S from E and a collection U' of subsets of S from V'; we then show that V' is a vertex semi-cover of G (solution of RVCP) if and only if U' is a cover of S (solution of SCP).

Suppose the vertices of V are labelled $1, 2, \ldots, n$ and the edges (u, v), $u < v$, of E are labelled uv. Let S be the set of all labels of edges of E. Let $V' = \{i_1, i_2, \ldots, i_r\}$. We construct U' (initially empty) according to the following steps, each performed for every $j \in 1..r$:

1. Determine $N(i_j)$, the set of vertices adjacent to (in the neighbourhood of) vertex i_j.
2. Form S_j, the set of edge labels $i_j q$ $(i_j < q)$ or $q i_j$ $(q < i_j)$ identified by the vertices $q \in N(i_j)$.
3. For every $\ell \neq j$, form $S_{j\ell}$, the set of edge labels pq $(p < q)$ or qp $(q < p)$ determined by all vertices $p \in N(i_j)$ that are adjacent to vertices $q \in N(i_\ell)$.
4. Form $U' \leftarrow U' \cup S_j \cup \big(\cup_\ell S_{j\ell}\big)$.

By construction, we see that V' is a semi-cover of G if and only if U' is a cover of S. $\square$

As noted in the introduction, there exists an efficient algorithm that computes solutions of SCP to within a logarithmic factor of problem size [J74,L75]. By

Theorem 3 the same is true for RVCP, and so we can compute, in addition to (2),

$$\nu_{RVC}^{EST} \in O(\log n \times \nu_{RVC}^{*}). \tag{4}$$

Of course we would really like to have the analogue of (2):

$$\nu_{RVC}^{EST} \in \nu_{RVC}^{*}..2\nu_{RVC}^{*} \ ??? \tag{5}$$

In the next section we discuss this open question further.

We conclude this section with the following observations:

- Suppose that the substrings $\boldsymbol{u_i}$ of U_ν are no longer constrained to have length k but may instead have any positive length $\leq k$. This change leads to the idea of a $\leq k$**-cover**, say U'_ν, and a corresponding problem (P1$'$). The graph $G_{\boldsymbol{x},k}$ can be formed in the same way as before by executing Algorithm $A(\boldsymbol{x}, k')$ for every $k' \in 2..k$, while eliminating any duplicate edges that result during the construction process. If we call the new algorithm $A'(\boldsymbol{x}, k)$, then an analogue to Theorem 1 holds: (P1$'$) is equivalent to RVCP restricted to graphs formed according to $A'(\boldsymbol{x}, k)$.
- Observe that, by virtue of the way in which edges are formed in $G_{\boldsymbol{x},k}$, all the edges must be expressed in the form (u, v), $u < v$. Thus if we interpret the edges as arcs in a directed graph $\overline{G_{\boldsymbol{x},k}}$, the set $\mathcal{G}$ of all such digraphs $\overline{G_{\boldsymbol{x},k}}$ is therefore a subset of the set of all acyclic digraphs. $\mathcal{G}$ is moreover a proper subset since none of its members can be a tournament.

3 Open Problems

We have shown that for $k \geq 2$, the k-cover problem kCP is equivalent to a restriction of RVCP, hence that efficient algorithms can be used to approximate a minimum k-cover as specified by (3) and (4). Interesting questions remain:

(Q1) The main outstanding question is whether or not (5) holds for RVCP, hence by Theorem 1 for kCP. For the closely-related problem VCP, one standard algorithm [S82] for computing $\nu_{VC}^{EST} \leq 2\nu_{VC}^{*}$ is based on ***depth first search*** (DFS):

compute a DFS tree T_G of G;
$V' \leftarrow$ the set of internal nodes of T_G.

For RVCP an analogous algorithm would need to include in V' only every *other* level of the internal nodes; that is,

- parents P_1 of leaf nodes L in T_G;
- parents P_3 of parents of P_1 in T_G;
- etc.

But such an algorithm can yield a very bad estimate of ν_{RVC}^{*}; for example, DFS from source vertex 1 of the graph

$$V = \{1, 2, \ldots, 2r{+}2\};$$
$$E = \big\{(1, 2);\ (2, i), 1 \leq i \leq 2r{+}2;\ (i{+}2, i{+}r{+}2), 1 \leq i \leq r\big\}$$

can lead to a covering set $V' = \{1, 3, 4, \ldots, r+2\}$ of cardinality $r+1$, while in fact $\nu^*_{RVC} = 1$ for this graph. A similar example exists for ***breadth first search*** (BFS).

It appears that in order to compute a good estimate of ν^*_{RVC}, the vertices of maximum degree should be given special consideration. The following greedy algorithm for graphs G is a generalization of one of the approximation algorithms proposed for strings $\boldsymbol{x}$ in [Y00]:

> $V' \leftarrow 0$;
> **while** $E \neq \emptyset$ **do**
>> find a vertex v of maximum degree in G;
>> $V' \leftarrow V' \cup \{v\}$;
>> remove v and all its incident edges from G.

This algorithm can be implemented in $O(n \log n)$ time, where $|V| = n$, and we conjecture that it computes $|V'| \leq 2\nu^*_{RVC}$, in accordance with (5).

(Q2) The set $\mathcal{G}$ of graphs $G_{\boldsymbol{x},k}$ formed by Algorithm $A(\boldsymbol{x}, k)$ in some sense describes the structure of all strings. To our knowledge these graphs have not previously been reported in the literature. Can the graphs of $\mathcal{G}$ be characterized in another way? What are their defining properties?

(Q3) The NP-completeness proof given in [CIMSY03] is based upon strings whose length n is a function of three parameters: k (the length of the covering substrings), r (the number of variables in the corresponding 3-SAT problem), and s (the number of clauses in the corresponding 3-SAT problem). A short calculation shows that in fact

$$n = (18k+7)r + (42k-3)s + (2k-1),$$

while at the same time the minimum cover size

$$\nu = 9r + 6r' + 8s + 1,\ r' \leq r.$$

Let us call the ratio $\gamma_k = n/(\nu k)$ the ***k-coverability*** of the string $\boldsymbol{x}[1..n]$; observe that γ_k has as an upper bound the average number of occurrences in $\boldsymbol{x}$ of the strings in the minimum k-cover. Since $\nu \leq 15r+8s+1$, we see then that for the class of strings constructed in [CIMSY03], $\gamma_k > 6/5$; in other words, the strings in the k-cover occur on average somewhat frequently in $\boldsymbol{x}$. What happens when $\gamma_k \leq 6/5$? Can we find a polynomial-time algorithm to compute a minimum k-cover given that γ_k falls below a certain threshold? For "most" strings and some sufficiently large k, we expect that $\nu = \lceil n/k \rceil$, so that $\gamma_k \approx 1$; thus such an algorithm would in fact handle most of the cases that arise.

References

[AHU74] Alfred V. Aho, John E. Hopcroft & Jeffey D. Ullman, *The Design & Analysis of Computer Algorithms*, Addison-Wesley (1974).

[AFI91] A. Apostolico, M. Farach and C. S. Iliopoulos, **Optimal superprimitivity testing for strings**, *Inform. Process. Lett.39* (1991) 17–20.

[AP83] Alberto Apostolico & Franco P. Preparata, **Optimal off-line detection of repetitions in a string**, *TCS 22* (1983) 297–315.

[BLPS00] Gerth S. Brodal, Rune B. Lyngsø, Christian N. S. Pedersen & Jens Stoye, **Finding maximal pairs with bounded gap**, *J. Discrete Algs. 1* (2000) 77–103.

[CIMSY03] Richard Cole, Costas S. Iliopoulos, Manal Mohamed, W. F. Smyth & Lu Yang, **Computing the minimum k-cover of a string**, *Proc. Prague Stringology Conf. '03* (2003) 51–62.

[C81] Maxime Crochemore, **An optimal algorithm for computing all the repetitions in a word**, *Inform. Process. Lett.12–5* (1981) 244–248.

[FST03] František Franěk, W. F. Smyth & Yudong Tang, **Computing all repeats using suffix arrays**, *J. Automata, Languages & Combinatorics 8–4* (2003) to appear.

[GJ79] Michael R. Garey & David S. Johnson, *Computing and intractability: a guide to the theory of NP-completeness*, Freeman (1979).

[IS98] Costas S. Iliopoulos & W. F. Smyth, **On-line algorithms for k-covering**, *Proc. Ninth Australasian Workshop on Combinatorial Algorithms* (1998) 107–116.

[J74] David S. Johnson, **Approximation algorithms for combinatorial problems**, J. Computer & System Sciences 9–3 (1974) 256–278.

[KK00] Roman Kolpakov & Gregory Kucherov, **On maximal repetitions in words**, *J. Discrete Algs. 1* (2000) 159–186.

[LS02] Yin Li and W. F. Smyth, **Computing the cover array in linear time**, *Algorithmica 32–1* (2002) 95–106.

[L75] László Lovász, **On the ratio of optimal integral and fractional covers**, *Discrete Math. 13* (1975) 383-390.

[M89] Michael G. Main, **Detecting leftmost maximal periodicities**, *Discrete Applied Maths. 25* (1989) 145–153.

[ML84] Michael G. Main & Richard J. Lorentz, **An $O(n \log n)$ algorithm for finding all repetitions in a string**, *J. Algs. 5* (1984) 422–432.

[S82] Carla D. Savage, **Depth first search and the vertex cover problem**, *Inform. Process. Lett.14* (1982) 233–237.

[S98] Jeanette P. Schmidt, **All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings**, *SIAM J. Comput. 27–4* (1998) 972–992.

[Y00] Lu Yang, *Computing the Minimum k-Cover of a String*, M. Sc. thesis, McMaster University (2000) 86 pp.