The Closest Substring problem with small distances*

Dániel Marx
Department of Computer Science and Information Theory,
Budapest University of Technology and Economics
Budapest H-1521, Hungary
dmarx@cs.bme.hu

Abstract

In the CLOSEST SUBSTRING *problem k strings $s_1, \ldots, s_k$ are given, and the task is to find a string s of length L such that each string s_i has a consecutive substring of length L whose distance is at most d from s. The problem is motivated by applications in computational biology. We present two algorithms that can be efficient for small fixed values of d and k: for some functions f and g, the algorithms have running time $f(d) \cdot n^{O(\log d)}$ and $g(d,k) \cdot n^{O(\log\log k)}$, respectively. The second algorithm is based on connections with the extremal combinatorics of hypergraphs. The* CLOSEST SUBSTRING *problem is also investigated from the parameterized complexity point of view. Answering an open question from [6, 7, 11, 12], we show that the problem is* W[1]*-hard even if both d and k are parameters. It follows as a consequence of this hardness result that our algorithms are optimal in the sense that the exponent of n in the running time cannot be improved to $o(\log d)$ or to $o(\log\log k)$ (modulo some complexity-theoretic assumptions). Another consequence is that the running time $n^{O(1/\varepsilon^4)}$ of the approximation scheme for* CLOSEST SUBSTRING *presented in [13] cannot be improved to $f(\varepsilon) \cdot n^c$, i.e., the ε has to appear in the exponent of n.*

1 Introduction

In this paper we are investigating a pattern matching problem that received considerable attention lately. Given k strings $s_1, \ldots, s_k$ over an alphabet Σ, and two integers L, d, the CLOSEST SUBSTRING problem asks whether there is a length L string s such that every string s_i has a length L substring s_i' whose Hamming-distance is at most d from s. The problem is motivated by applications in computational biology. Finding similar regions in multiple DNA, RNA, or protein sequences plays an important role in many applications, for example, in locating binding sites and in finding conserved regions in unaligned sequences.

The CLOSEST SUBSTRING problem is NP-hard even in the special case when $\Sigma = \{0,1\}$ and every string s_i has length L (cf. [9]). Li et. al [13] studied the optimization version of CLOSEST SUBSTRING, where we have to find the smallest d that makes the problem feasible. They presented a polynomial-time approximation scheme: for every $\varepsilon > 0$, there is an $n^{O(1/\varepsilon^4)}$ time algorithm that produces a solution that is at most $(1+\varepsilon)$-times worse than the optimum.

Parameterized complexity deals with NP-hard problems where every instance has a distinguished part k, which will be called the parameter. We expect that for an NP-hard problem every algorithm has exponential running time. In parameterized complexity the goal is to develop algorithms that run in *uniformly polynomial time*: the running time is $f(k) \cdot n^c$, where c is a constant and f is a (possibly exponential) function depending only on k. We call a parameterized problem *fixed-parameter tractable* if such an algorithm exists. This means that the exponential increase of the running time can be restricted to the parameter k. It turns out that several NP-hard problems are fixed-parameter tractable, for example MINIMUM VERTEX COVER, LONGEST PATH, and DISJOINT TRIANGLES. Therefore, for small values of k, the $f(k)$ term is just a constant factor in the running time, and the algorithms for these problems can be efficient even for large values of n. This has to be contrasted with algorithms that have running time such as n^k: in this case the algorithm becomes practically useless for large values of n even if k is as small as 10. The theory of W[1]-hardness can be used to show that a problem is unlikely to be fixed-parameter tractable, for every algorithm the parameter has to appear in the exponent of n. For example, for MAXIMUM CLIQUE and MINIMUM DOMINATING SET the running time of the best known algorithms is $n^{O(k)}$, and the W[1]-hardness of these problems tells us that it is unlikely that an algorithm with running time, say, $O(2^k \cdot n)$ can be found. For more details, see [5].

*Research is supported in part by grants OTKA 44733, 42559 and 42706 of the Hungarian National Science Fund.

CLOSEST SUBSTRING was investigated in the framework of parameterized complexity by several authors. Formally, the problem is the following:

> CLOSEST SUBSTRING
>
> *Input:*
> k strings $s_1, \ldots, s_k$ over an alphabet Σ, integers d and L.
>
> *Parameters:*
> k, $|\Sigma|$, d, L
>
> *Task:*
> Find a string s of length L such that for every $1 \leq i \leq k$, the string s_i has a length L consecutive substring s_i' with $d(s, s_i') \leq d$.

The string s is called the *center string*. The Hamming-distance of two strings w_1 and w_2 (i.e., the number of positions where they differ) is denoted by $d(w_1, w_2)$. For a given center string s, it is easy to check in polynomial time whether the substrings s_i' exist: we have to try every length L substring of the strings s_i.

In [8] and [6] it is shown that the problem is W[1]-hard even if all three of k, d, and L are parameters. Therefore, if the size of the alphabet Σ is not bounded in the input, then we cannot hope for an efficient exact algorithm for the problem. However, in the computational biology applications the strings are typically DNA or protein sequences, hence the number of different symbols is a small constant. Therefore, we will focus on the case when the size of Σ is a parameter. Restricting $|\Sigma|$ does not make the problem tractable, since CLOSEST SUBSTRING is NP-hard even if the alphabet is binary. On the other hand, if $|\Sigma|$ and L are both parameters, then the problem becomes fixed-parameter tractable: we can enumerate and check all the $|\Sigma|^L$ possible center strings. However, in practical applications the strings are usually very long, hence it makes much more sense to restrict the number of strings k or the distance parameter d. In [7] it is shown that CLOSEST SUBSTRING is W[1]-hard with parameter k, even if the alphabet is binary. However, the complexity of the problem with parameter d or with combined parameters d, k remained an open question.

Our results. We show that CLOSEST SUBSTRING is W[1]-hard with combined parameters k and d, even if the alphabet is binary. This resolves an open question raised in [6, 7, 11, 12]. Therefore, there is no $f(k,d) \cdot n^c$ algorithm for CLOSEST SUBSTRING (unless FPT = W[1]); the exponential increase cannot be restricted to the parameters k and d. The first step in the reduction is to introduce a technical problem called SET BALANCING, and prove W[1]-hardness for this problem. This part of the proof contains most of the new combinatorial ideas. The SET BALANCING problem is reduced to CLOSEST SUBSTRING by a reduction very similar to the one presented in [7].

We present two exact algorithms for the CLOSEST SUBSTRING problem. These algorithms can be efficient if d, or both d and k are small (less than $\log n$). The first algorithm runs in $|\Sigma|^{d(\log d+2)} n^{O(\log d)}$ time. Notice that this algorithm is not uniformly polynomial, but only the logarithm of the parameter appears in the exponent of n. Therefore, the algorithm might be efficient for small values of d. The second algorithm has running time $(|\Sigma| d)^{O(kd)} \cdot n^{O(\log\log k)}$. Here the parameter k appears in the exponent of n, but $\log\log k$ is a very slowly growing function. This algorithm is based on defining certain hypergraphs and enumerating all the places where one hypergraph appears in the other. Using some results from extremal combinatorics, we develop techniques that can speed up the search for hypergraphs. It turns out that if hypergraph H has bounded fractional edge cover number, then we can enumerate in uniformly polynomial time all the places where H appears in some larger hypergraph G. This result might be of independent interest.

Notice that the running times of our two algorithms are incomparable. Assume that $|\Sigma| = 2$. If $d = O(\log n)$ and $k = n^{O(1)}$, then the running time of the first algorithm is $n^{O(\log\log n)} \cdot n^{O(\log\log n)} = n^{O(\log\log n)}$, while the second algorithm needs $(\log n)^{n^{O(1)} \log n} \cdot n^{O(\log\log n)}$ steps, which can be much larger. On the other hand, if $d = O(\log\log n)$ and $k = O(\log\log n)$, then the first algorithm runs in something like $n^{O(\log\log\log n)}$ time, while the second algorithm needs only $(\log\log n)^{O(\log^2 \log n)} \cdot n^{O(\log\log\log\log n)} = n^{O(\log\log\log\log n)}$ steps.

Our W[1]-hardness proof combined with some recent results on subexponential algorithms shows that the two exact algorithms are in some sense best possible. The exponents are optimal: we show that if there is an $f_1(k,d,|\Sigma|) \cdot n^{o(\log d)}$ or an $f_2(k,d,|\Sigma|) \cdot n^{o(\log\log k)}$ algorithm for CLOSEST SUBSTRING, then 3-SAT can be solved in subexponential time.

If a PTAS has running time such as $O(n^{1/\varepsilon^2})$, then it becomes practically useless for large n, even if we ask for an error bound of 20%. An *efficient PTAS (EPTAS)* is an approximation scheme that produces a $(1+\varepsilon)$-approximation in $f(\varepsilon) \cdot n^c$ time for some constant c. If $f(\varepsilon)$ is e.g., $2^{1/\varepsilon}$, then such an approximation scheme can be practical even for $\varepsilon = 0.1$ and large n. A standard consequence of W[1]-hardness is that there is no EPTAS for the optimization version of the problem. Hence our hardness result shows that the $n^{O(1/\varepsilon^4)}$ time approximation scheme of [13] for CLOSEST SUBSTRING cannot be improved to an EPTAS.

The paper is organized as follows. The first algorithm is presented in Section 2. In Section 3 we discuss techniques for finding one hypergraph in another. In Section 4 we present the second algorithm. This section introduces a new hypergraph property called half-covering, which plays an important role in the algorithm. We define in Section 5 the SET BALANCING problem, and prove that it is W[1]-hard. In Section 6 the SET BALANCING problem is used to

show that CLOSEST SUBSTRING is W[1]-hard with combined parameters d and k. We conclude the paper with a summary in Section 7.

2 Finding generators

In this section we present an algorithm with running time proportional to roughly $n^{\log d}$. The algorithm is based on the following observation: if all the strings $s'_1, \ldots, s'_k$ agree at some position p in the solution, then we can safely assume that the same symbol appears at the p-th position of the center string s. However, if we look at only a subset of the strings $s'_1, \ldots, s'_k$, then it is possible that they all agree at some position, but the center string contains a different symbol at this position. We will be interested in sets of strings that do not have this problem:

Definition 2.1. *Let $G = \{g_1, g_2, \ldots, g_\ell\}$ be a set of length L strings. We say that G is a* generator *of the length L string s if whenever every g_i has the same character at some position p, then string s has this character at position p. The* size *of the generator is ℓ, the number of strings in G. The* conflict *of the generator is the set of those positions where not all of the strings g_i have the same character.*

As we have argued above, it can be assumed that the strings $s'_1, \ldots, s'_k$ of a solution form a generator of the center string s. Furthermore, these strings have a subset of size at most $\log d + 2$ that is also a generator:

Lemma 2.2. *If an instance of* CLOSEST SUBSTRING *is solvable, then there is a solution s that has a generator G having the following properties:*

- *each string in G is a substring of some s_i,*
- *G has size at most $\log d + 2$,*
- *the conflict of G is at most $d(\log d + 2)$.*

Proof. Let $s, s'_1, \ldots, s'_k$ be a solution such that $\sum_{i=1}^{k} d(s, s'_i)$ is minimal. We prove by induction that for every j we can select a set G_j of j strings from $s'_1, \ldots, s'_k$ such that there are less than $(d+1)/2^{j-1}$ *bad positions* where the strings in G_j all agree, but this common character is different from the character in s at this position. The lemma follows from $j = \lceil \log(d+1) \rceil + 1 \leq \log d + 2$: the set G_j has no bad positions, hence it is a generator of s. Furthermore, each string in G_j is at distance at most d from s, thus the conflict of G_j can be at most $d(\log d + 2)$.

For the case $j = 1$ we can set $G_1 = \{s'_1\}$, since s'_1 differs from s at not more than d positions. Now assume that the statement is true for some j. Let P be the set of bad positions, where the j strings in G_j agree, but they differ from s. We claim that there is some string s'_t in the solution and a subset $P' \subseteq P$ with $|P'| > |P|/2$ such that s'_t differs from all the strings in G_j at every position of P'. If this is true, then we add s'_t to the set G_j to obtain G_{j+1}. Only the positions in $P \setminus P'$ are bad for the set G_{j+1}: for every position p in P', the strings cannot all agree at p, since s'_t do not agree with the other strings at this position. Thus there are at most $|P \setminus P'| < |P|/2 < (d+1)/2^j$ bad positions, completing the induction.

Assume that there is no such string s'_t. In this case we modify the center string s the following way: for every position $p \in P$, let the character at position p be the same as in string s'_1. Denote by s^* the new string. We show that $d(s^*, s'_i) \leq d(s, s'_i) \leq d$ for every $1 \leq i \leq k$, hence s^* is also a solution. By assumption, every string s'_i in the solution agrees with s'_1 on at least $|P|/2$ positions of P. Therefore, if we replace s with s^*, the distance of s'_i from the center string decreases on at least $|P|/2$ positions, and the distance can increase only on the remaining at most $|P|/2$ positions. Therefore, $d(s^*, s'_i) \leq d(s, s'_i)$ follows. Furthermore, $d(s^*, s'_1) = d(s, s'_1) - |P|$ implies $\sum_{i=1}^{k} d(s^*, s'_i) < \sum_{i=1}^{k} d(s, s'_i)$, which contradicts the minimality of s. □

Our algorithm first creates a set S containing all the length L substrings of $s_1, \ldots, s_k$. For every subset $G \subseteq S$ of $\log d + 2$ strings, we check whether G generates a center string s that solves the problem. Since $|S| \leq n$, there are at most $n^{\log d + 2}$ possibilities to try. By Lemma 2.2 we have to consider only those generators whose conflict is at most $d(\log d + 2)$, hence at most $|\Sigma|^{d(\log d+2)}$ possible center strings have to be tested for each G.

Theorem 2.3. CLOSEST SUBSTRING *can be solved in $|\Sigma|^{d(\log d+2)} n^{\log d + O(1)}$ time.* □

3 Finding hypergraphs

Let us recall some standard definitions concerning hypergraphs. A *hypergraph* $H(V_H, E_H)$ consists of a set of *vertices* V_H and a collection of *edges* E_H, where each edge is a subset of V_H. Let $H(V_H, E_H)$ and $G(V_G, E_G)$ be two hypergraphs. We say that H *appears at $V' \subseteq V_G$ as partial hypergraph* if there is a bijection π between the elements of V_H and V' such that for every edge $E \in E_H$ we have that $\pi(E)$ is an edge of G (where the mapping π is extended to the edges the obvious way). For example, if H has the edges $\{1,2\}$, $\{2,3\}$, and G has the edges $\{a,b\}$, $\{b,c\}$, $\{c,d\}$, then H appears as a partial hypergraph at $\{a,b,c\}$ and at $\{b,c,d\}$. We say that H *appears at $V' \subseteq V_G$ as subhypergraph* if there is such a bijection π where for every $E \in E_H$, there is an edge $E' \in E_G$ with $\pi(E) = E' \cap V'$. For example, let the edges of H be $\{1,2\}$, $\{2,3\}$, and let the edges of G be $\{a,c,d\}$, $\{b,c,d\}$. Now H does not appear in G as partial hypergraph, but H appears as subhypergraph at $\{a,b,c\}$ and at $\{a,b,d\}$. If H appears at some $V' \subseteq V_G$ as partial hypergraph, then it appears there as subhypergraph as well.

A *stable set* in $H(V_H, E_H)$ is a subset $S \subseteq V_H$ such that every edge of H contains at most one element from S. The *stable number* $\alpha(H)$ is the size of the largest stable set in H. A *fractional stable set* is an assignment $\phi\colon V_H \to [0,1]$ such that $\sum_{v\in E}\phi(v) \le 1$ for every edge E of H. The *fractional stable number* $\alpha^*(H)$ is the maximum of $\sum_{v\in V_H}\phi(v)$ taken over all fractional stable sets ϕ. The incidence vector of a stable set is a fractional stable set, hence $\alpha^*(H) \ge \alpha(H)$. An *edge cover* of H is a subset $E' \subseteq E_H$ such that each vertex of V_H is contained in at least one edge of E'. The *edge cover number* $\rho(H)$ is the size of the smallest edge cover in H. (The hypergraphs considered here do not have isolated vertices, hence every hypergraph has an edge cover.) A *fractional edge cover* is an assignment $\psi\colon E_H \to [0,1]$ such that $\sum_{E\ni v}\psi(E) \ge 1$ for every vertex v. The *fractional cover number* $\rho^*(H)$ is the minimum of $\sum_{E\in E_H}\psi(E)$ taken over all fractional edge covers ψ, clearly $\rho^*(H) \le \rho(H)$. It follows from the duality theorem of linear programming that $\alpha^*(H) = \rho^*(H)$ for every hypergraph H.

Friedgut and Kahn [10] determined the maximum number of times a hypergraph $H(V_H, E_H)$ can appear as partial hypergraph in a hypergraph G with m edges. That is, we are interested in the maximum number of different subsets $V' \subseteq V_G$ where H can appear in G. A trivial upper bound is $m^{|E_H|}$: if we fix $\pi(E) \in E_G$ for each edge $E \in E_H$, then this uniquely determines $\pi(V_H)$. This bound can be improved to $m^{\rho(H)}$: if edges $E_1, E_2, \ldots, E_{\rho(H)}$ cover every vertex of V_H, then by fixing $\pi(E_1)$, $\pi(E_2)$, $\ldots$, $\pi(E_{\rho(H)})$ the set $\pi(V_H)$ is determined. The result of Friedgut and Kahn says that ρ can be replaced with the (possibly smaller) ρ^*:

Theorem 3.1 ([10]). *Let H be a hypergraph with fractional cover number $\rho^*(H)$, and let G be a hypergraph with m edges. The maximum number of times H can appear in G as partial hypergraph is $|V_H|^{|V_H|}\cdot m^{\rho^*(H)}$. Furthermore, for every H and sufficiently large m, there is a hypergraph with m edges where H appears $m^{\rho^*(H)}$ times.*

Theorem 3.1 does not remain valid if we replace "partial hypergraph" with "subhypergraph." For example, let H contain only one edge $\{1,2\}$, and let G have one edge E of size ℓ. Now H appears at each of the $\binom{\ell}{2}$ two element subsets of E as subhypergraph. However, if we bound the size of the edges in G, then we can state a subhypergraph analog of Theorem 3.1:

Corollary 3.2. *Let H be a hypergraph with fractional cover number $\rho^*(H)$, and let G be a hypergraph with m edges, each of size at most ℓ. Hypergraph H can appear in G as subhypergraph at most $|V_H|^{|V_H|}\cdot \ell^{|V_H|\rho^*(H)}\cdot m^{\rho^*(H)}$ times.* □

Given hypergraphs $H(V_H, E_H)$ and $G(V_G, E_G)$, we would like to find all the places $V' \subseteq V_G$ in G where H appears as subhypergraph. By Corollary 3.2, there can be at most $t = |V_H|^{|V_H|}\cdot \ell^{|V_H|\rho^*}\cdot m^{\rho^*}$ such places, which means that we cannot enumerate all of them in less than $\Theta(t)$ steps. Therefore, our aim is to find an algorithm with running time polynomial in t. The proof of Theorem 3.1 is not algorithmic (it is based on Shearer's Lemma [4], which is proved by entropy arguments), hence it does not directly imply an efficient way of enumerating all the places where H appears. However, in Theorem 3.3, we show that there is a very simple algorithm for enumerating all these places. Corollary 3.2 is used to bound the running time of the algorithm. This result might be useful in other applications as well.

Theorem 3.3. *Let $H(V_H, E_H)$ be a hypergraph with fractional cover number $\rho^*(H)$, and let $G(V_H, E_H)$ be a hypergraph with m edges where each edge has size at most ℓ. There is an algorithm that enumerates in $|V_H|!\cdot|V_H|^{|V_H|}\cdot \ell^{|V_H|\rho^*(H)}\cdot m^{\rho^*(H)+O(1)}$ time every subset $V' \subseteq V_G$ where H appears in G as subhypergraph.*

Proof. Let $V_H = \{1,2,\ldots,r\}$. For each $1 \le i \le r$, let H_i be a hypergraph on $V_i = \{1,2,\ldots,i\}$ such that if E is an edge of H, then $E \cap V_i$ is an edge of H_i. For each $i = 1,2,\ldots,r$, we find all the places where H_i appears in G as subhypergraph. Since $H = H_r$ this method will solve the problem.

For $i = 1$ the problem is trivial, since V_i has only one vertex. Assume now that we have a list L_i of all the i element subsets of V_G where H_i appears as subhypergraph. The important observation is that if H_{i+1} appears as subhypergraph at some $V' \subseteq V_G$, then V' has an i element subset V'' where H_i appears as subhypergraph. For each set $X \in L_i$, we try all the $|V_G \setminus X|$ different ways of extending X to an $i+1$ element set X', and check whether H_{i+1} appears at X' as subhypergraph. This can be checked by trying all the $(i+1)!$ possible bijections π between V_{i+1} and X', and by checking for each edge E of H_{i+1} whether there is an edge E' in G with $\pi(E) = E' \cap X'$.

Let us estimate the running time of the algorithm. The algorithm consists of $|V_H|$ iterations. Notice first that $\rho^*(H_i) \le \rho^*(H)$, since a fractional edge cover of H can be used to obtain a fractional edge cover of H_i. Therefore, by Corollary 3.2, each list L_i has size at most $|V_H|^{|V_H|}\cdot \ell^{|V_H|\rho^*(H)}\cdot m^{\rho^*(H)}$. When we determine the list L_{i+1}, we have to check for at most $|L_i|\cdot|V_G|$ different size $i+1$ sets X' whether H_{i+1} appears at X' as subhypergraph. Checking one X' requires us to test $(i+1)!$ different bijections π, and for each π we have to go through all the m edges of G. Suppressing the polynomial factors, the total running time is $|V_H|!\cdot|V_H|^{|V_H|}\cdot \ell^{|V_H|\rho^*(H)}\cdot m^{\rho^*(H)+O(1)}$. □

4 Half-covering and the CLOSEST SUBSTRING problem

The following hypergraph property plays a crucial role in our second algorithm for CLOSEST SUBSTRING:

Definition 4.1. *We say that a hypergraph $H(V,E)$ has the* half-covering *property if for every non-empty subset $Y \subseteq V$ there is an edge $X \in E$ with $|X \cap Y| > |Y|/2$.*

Theorem 3.3 says that finding a hypergraph H is easy if H has small fractional cover number. In our algorithm for the CLOSEST SUBSTRING problem (described later in this section), we have to find hypergraphs satisfying the half-covering property. The following combinatorial lemma shows that such hypergraphs have small fractional cover number, hence they are easy to find:

Lemma 4.2. *If $H(V,E)$ is a hypergraph with m edges satisfying the half-covering property, then the fractional cover number ρ^* of H is $O(\log\log m)$.*

Proof. The fractional cover number equals the fractional stable number, thus there is a function ϕ: $V \to [0,1]$ such that $\sum_{v\in X}\phi(v) \le 1$ holds for every edge $X \in E$, and $\sum_{v\in V}\phi(v) = \rho^*$. Let $v_1, v_2, \ldots, v_{|V|}$ be an ordering of the vertices by decreasing value of $\phi(v_i)$. First we give a bound on the sum of the largest $\phi(v_i)$'s:

Proposition 4.3. *For every $1 \le i \le |V|$, we have $\sum_{j=1}^{i}\phi(v_j) \le -4\log_2\phi(v_i) + 4$.*

Proof. The proof is by induction on i. Since $\phi(v_1) \le 1$, the claim is trivial for $i = 1$. For an arbitrary $i > 1$, let $i' \le i$ be the smallest value such that $\phi(v_{i'}) \le 2\phi(v_i)$. By assumption, there is an edge X of H that covers more than half of the set $S = \{v_{i'}, \ldots, v_i\}$. Every weight in S is at least $\phi(v_i)$, hence X can cover at most $1/\phi(v_i)$ elements of S. Thus $|S| \le 2/\phi(v_i)$, and $\sum_{j=i'}^{i}\phi(v_j) \le 4$ follows from the fact that $\phi(v_j) \le 2\phi(v_i)$ for $i' \le j \le i$. If $i' = 1$, then we are done. Otherwise $\sum_{j=1}^{i'-1}\phi(v_j) \le -4\log_2\phi(v_{i'-1}) + 4 < -4(\log_2\phi(v_i)+1)+4$ follows from the induction hypothesis and from $\phi(v_{i'-1}) > 2\phi(v_i)$. Therefore, $\sum_{j=1}^{i}\phi(v_j) = \sum_{j=1}^{i'-1}\phi(v_j) + \sum_{j=i'}^{i}\phi(v_j) \le -4\log_2\phi(v_i)+4$, what we had to show. □

In the rest of the proof we assume that ρ^* is sufficiently large, say $\rho^* \ge 100$. Let i be the largest value such that $\sum_{j=i}^{|V|} \ge \rho^*/2$. By the definition of i, $\sum_{j=i+1}^{|V|}\phi(v_j) < \rho^*/2$, hence $\sum_{j=1}^{i}\phi(v_j) \ge \rho^*/2$. Thus by Prop. 4.3, the weight of v_i (and every v_j with $j \ge i$) is at most $2^{-(\rho^*/2-4)/4} \le 2^{-\rho^*/10}$ (assuming that ρ^* is sufficiently large). Define $T := \{v_i, \ldots, v_{|V|}\}$, and let us select a random subset $Y \subseteq T$: independently each vertex $v_j \in T$ is selected into Y with probability $p(v_j) := 2^{\rho^*/10}\cdot\phi(v_j) \le 1$. We show that if H does not have $2^{2^{\Omega(\rho^*)}}$ edges, then with nonzero probability every edge of H covers at most half of Y, contradicting the assumption that H satisfies the half-covering property.

The size of Y is the sum of $|T|$ independent 0-1 random variables. The expected value of this sum is $\mu = \sum_{j=i}^{|V|}p(v_j) = 2^{\rho^*/10}\cdot\sum_{j=i}^{|V|}\phi(v_j) \ge 2^{\rho^*/10}\cdot\rho^*/2$. We show that with nonzero probability $|Y| > \mu/2$, but $|X\cap Y| < \mu/4$ for every edge X. To bound the probability of the bad events, we use the following form of the Chernoff Bound:

Theorem 4.4 ([1]). *Let $X_1, X_2, \ldots, X_n$ be independent* 0-1 *random variables with $\Pr[X_i = 1] = p_i$. Denote $X = \sum_{i=1}^{n}X_i$ and $\mu = \mathrm{E}[X]$. Then*

$$\Pr[X \le (1-\beta)\mu] \le \exp(-\beta^2\mu/2) \quad \text{for } 0 < \beta \le 1,$$
$$\Pr[X \ge (1+\beta)\mu] \le \begin{cases} \exp(-\beta^2\mu/3) \text{ for } 0 < \beta \le 1, \\ \exp(-\beta^2\mu/(2+\beta)) \text{ for } \beta > 1. \end{cases}$$

Thus by setting $\beta = \frac{1}{2}$, the probability that Y is too small can be bounded as

$$\Pr[|Y| \le \mu/2] \le \exp(-1/8\mu).$$

For each edge X, the random variable $|X\cap Y|$ is the sum of $|X\cap T|$ independent 0-1 random variables. The expected value of this sum is $\mu_X = \sum_{v\in X\cap T}p(v) = 2^{\rho^*/10}\cdot\sum_{v\in X\cap T}\phi(v) \le 2^{\rho^*/10} \le \mu/(\rho^*/2)$, where the first inequality follows from the fact that ϕ is a fractional stable set, hence the total weight X can cover is at most 1. Notice that if ρ^* is sufficiently large, than the expected size of $X\cap Y$ is much smaller than the expected size of Y. We want to bound the probability that $|X\cap Y|$ is at least $\mu/4$. Setting $\beta = (\mu/4)/\mu_X - 1 \ge \rho^*/8 - 1$, the Chernoff Bound gives

$$\begin{aligned}\Pr\big[|X\cap Y| \ge \mu/4\big] &= \Pr\big[|X\cap Y| \ge (1+\beta)\mu_X\big] \\ &\le \exp(-\beta^2\mu_X/(2+\beta)) \le \exp(-\beta^2\mu_X/(2\beta)) = \\ &\qquad \exp(-\mu/8 + \mu_X/2) \le \exp(-\mu/16).\end{aligned}$$

Here we assumed that ρ^* is sufficiently large that $\beta \ge 2$ (second inequality) and $\mu_X/2 \le \mu/16$ (third inequality) hold. If H has m edges, then the probability that $|Y| \le \mu/2$ holds or an edge X covers at least $\mu/4$ vertices of Y is at most

$$\begin{aligned}&\exp(-\mu/8) + m\cdot\exp(-\mu/16) \\ &\qquad \le (m+1)\exp(-2^{\rho^*/10}\cdot\rho^*/32) \le m\cdot 2^{-2^{\Omega(\rho^*)}}. \quad (1)\end{aligned}$$

If H satisfies the half-covering property, then for every Y there has to be at least one edge that covers more than half of Y. Therefore, the upper bound (1) has to be at least 1. This is only possible if m is $2^{2^{\Omega(\rho^*)}}$, and it follows that $\rho^* = O(\log\log m)$, what we had to show. □

We remark that the $O(\log\log m)$ bound in Lemma 4.2 is tight: one can construct a hypergraph satisfying the half-covering property that has fractional cover number k and 2^{2^k} edges.

Now we are ready to prove the main result of this section:

Theorem 4.5. CLOSEST SUBSTRING *can be solved in* $(|\Sigma|d)^{O(kd)} \cdot n^{O(\log\log k)}$ *time.*

Proof. Let us fix the first substring $s'_1 \in s_1$ of the solution. We will repeat the following algorithm for each possible choice of s'_1. Since there are at most n possibilities for choosing s'_1, the running time of the algorithm presented below has to be multiplied by a factor of n, which is dominated by the $n^{O(\log\log k)}$ term.

The center string s can differ on at most d positions from s'_1. Therefore, if we can find the set P of these positions, then the problem can be solved by trying all the $|\Sigma|^{|P|} \leq |\Sigma|^d$ possible assignments to the positions in P. We show how to enumerate efficiently all the possible sets P.

We construct a hypergraph G over the vertex set $\{1,\ldots,L\}$. The edges of the hypergraph describe the possible substrings in the solution. If w is a length L substring of some string s_i, then we add an edge E to G such that $p \in E$ if and only if the p-th character of w differs from the p-th character of s'_1. If $(s, s'_1, \ldots, s'_k)$ is a solution, then let H be the partial hypergraph of G that contains only the $k-1$ edges corresponding to the $k-1$ substrings $s'_2, \ldots, s'_k$. (H can have less than $k-1$ edges if the same edge corresponds to two different substrings.) Denote by P the set of at most d positions where s and s'_1 differ. Let H_0 be the subhypergraph of H induced by P: the vertex set of H_0 is P, and for each edge E of H there is an edge $E \cap P$ in H_0. Hypergraph H_0 is subhypergraph of H and H is partial hypergraph of G, thus H_0 appears in G at P as subhypergraph.

We say that a solution is *minimal* if $\sum_{i=1}^{k} d(s, s'_i)$ is minimal. In Prop. 4.6, we show that if the solution $(s, s'_1, \ldots, s'_k)$ is minimal, then H_0 has the half-covering property. Therefore, we can enumerate all the possible P's by considering every hypergraph H_0 on at most d vertices that has the half-covering property (there are only a constant number of them), and for each such H_0, we enumerate all the places in G where H_0 appears as subhypergraph. Lemma 4.2 ensures that every H_0 considered has small fractional cover number. By Lemma 3.3, this means that we can enumerate efficiently all the places P where H_0 appears in G as subhypergraph. As discussed above, for each such P we can check whether there is a solution where the center string s differs from s'_1 only on P. By repeating this method for every hypergraph H_0 having the half-covering property, we eventually find a solution, if exists.

Proposition 4.6. *For every minimal solution* $(s, s'_1, \ldots, s'_k)$*, the corresponding hypergraph* H_0 *has the half-covering property.*

Proof. To see that H_0 has the half-covering property, assume that for some $Y \subseteq P$, every edge of H_0 covers at most half of Y. We show that in this case the solution is not minimal. Modify s such that it is the same as s'_1 on every position of Y, let s^* be the new center string. Clearly, $d(s^*, s'_1) = d(s, s'_1) - |Y|$. Furthermore, we show that this modification does not increase the distance for any i, that is, $d(s^*, s'_i) \leq d(s, s'_i)$ for every i. This means that s^* is also a good center string, contradicting the minimality of the solution.

Let E_i be the edge of H_0 corresponding to the substring s'_i. This means that s'_1 and s'_i differ on $Y \cap E_i$, and they are the same on $Y \setminus E_i$. Therefore, $d(s^*, s'_i) \leq d(s, s'_i) + |Y \cap E_i| - |Y \setminus E_i|$. By assumption, E_i can cover at most half of Y, hence $d(s^*, s'_i) \leq d(s, s'_i)$, as required. □

The most important factor of the running time comes from using Theorem 3.3 to find all the places where H_0 appears in G as subhypergraph. Since H_0 satisfies the half-covering property and has less than k edges, by Lemma 4.2 its fractional covering number is $O(\log\log k)$. Therefore, the algorithm of Theorem 3.3 runs in roughly $n^{O(\log\log k)}$ time. The other factors of the running time (trying every possible H_0, checking every s corresponding to a given P, etc.) depends only on k, d, and Σ. □

5 Set Balancing

In this section we introduce a new problem called SET BALANCING. The problem is somewhat technical, it is not motivated by practical applications. However, as we will see it in Section 6, the problem is useful in proving the W[1]-hardness of CLOSEST SUBSTRING.

SET BALANCING

Input:
A collection of m set systems $\mathscr{S}_i = \{S_{i,1}, \ldots, S_{i,|\mathscr{S}_i|}\}$ $(1 \leq i \leq m)$ over the same ground set A and a positive integer d. The size of each set $S_{i,j}$ is at most ℓ, and there is an integer weight $w_{i,j}$ associated to each set $S_{i,j}$.

Parameters:
m, d, ℓ

Task:
Find a set $X \subseteq A$ of size at most d and select a set $S_{i,a_i} \in \mathscr{S}_i$ for every $1 \leq i \leq m$ in such a way that

$$|X \triangle S_{i,a_i}| \leq w_{i,a_i} \qquad (2)$$

holds for every $1 \leq i \leq m$.

Here $X \triangle S_{i,a_i}$ denotes the symmetric difference $|(X \setminus S_{i,a_i}) \cup (S_{i,a_i} \setminus X)|$. We have to select a set X and a set from each set system in such a way that the balancing requirement (2) is satisfied: every selected set is close to X. The weight $w_{i,j}$ of each set $S_{i,j}$ prescribes the maximum distance

of X from this set. The smaller the weight, the more restrictive the requirement. The distance is measured by symmetric difference; therefore, adding to X an element outside $S_{i,j}$ can be compensated by adding to X an element from $S_{i,j}$. If (2) holds for some set S_{i,a_i}, then we say that S_{i,a_i} is *balanced*, or X *balances* S_{i,a_i}.

It can be assumed that the weight of each set is at most $\ell + d$, otherwise the requirement would be automatically satisfied for every possible X. If a set appears in multiple set systems, then it can have different weights in the different systems.

Theorem 5.1. SET BALANCING *is* W[1]*-hard with parameters m, d, and ℓ.*

Proof. The proof is by reduction from the MAXIMUM CLIQUE problem. Assume that a graph $G(V,E)$ is given with n vertices and e edges, the task is to find a clique of size t. It can be assumed that $n = 2^{2^C}$ for some integer C: we can ensure that the number of vertices has this form by adding at most $|V|^2$ isolated vertices. Furthermore, we can assume that $C \geq t$ (i.e., $n \geq 2^{2^t}$): if $n < 2^{2^t}$, then MAXIMUM CLIQUE can be solved directly in time $(2^{2^t})^t \cdot n$ by enumerating every set of size t.

The ground set A of the SET BALANCING problem is partitioned into t groups $A_0, \ldots, A_{t-1}$. The group A_i is further partitioned into 2^i blocks $A_{i,1}, \ldots, A_{i,2^i}$; the total number of blocks is $2^t - 1$. The block $A_{i,j}$ contains $n^{1/2^i} = 2^{2^{C-i}}$ elements. Set $d := 2^t - 1$. Later we will argue that it is sufficient to restrict our attention to solutions where X contains exactly one element from each block $A_{i,j}$. Let us call such a solution a *standard solution*. We construct the set systems in such a way that there is one-to-one correspondence between the standard solutions and the size t cliques of G. In a standard solution X contains exactly 2^i elements from group A_i, and there are $(n^{1/2^i})^{2^i} = n$ different possibilities for selecting these 2^i elements from the blocks of A_i. Let the set system $\mathscr{X}_i = \{X_{i,1}, \ldots, X_{i,n}\}$ contain these n different 2^i element sets. These n possibilities will correspond to the choice of the i-th vertex of the clique.

The set systems are of two types: the verifier systems and the enforcer systems. The role of the verifier systems is to ensure that every standard solution corresponds to a clique of size t, while the enforcer systems ensure that there are only standard solutions.

For each $0 \leq i_1 < i_2 \leq t-1$ the verifier system $\mathscr{S}_{i_1,i_2}$ ensures that the i_1-th and the i_2-th vertices of the clique are adjacent. The set system $\mathscr{S}_{i_1,i_2}$ contains $2e$ sets of size $2^{i_1} + 2^{i_2}$ each. If vertices u and v are adjacent in G, then $X_{i_1,u} \cup X_{i_2,v}$ is in $\mathscr{S}_{i_1,i_2}$. The weight of every set in $\mathscr{S}_{i_1,i_2}$ is $(2^t - 1) - (2^{i_1} + 2^{i_2})$.

Proposition 5.2. *There is a standard solution if and only if G has a k-clique.*

Proof. Assume that $v_0, \ldots, v_{t-1}$ is a clique in G. Let

$$X = \bigcup_{i=0}^{t-1} X_{i,v_i}.$$

The size of X is $\sum_{i=0}^{t-1} 2^i = 2^t - 1$. Select the set $X_{i_1,v_{i_1}} \cup X_{i_2,v_{i_2}}$ from the verifier system $\mathscr{S}_{i_1,i_2}$. This set is balanced: it is a size $2^{i_1} + 2^{i_2}$ subset of X having weight $(2^t - 1) - (2^{i_1} + 2^{i_2})$.

To prove the other direction, assume now that there is a standard solution X. In a standard solution $X \cap A_i$ is a 2^i element set from $\mathscr{X}_i$, assume that $X \cap A_i = X_{i,v_i}$ for some v_i. We claim that the v_i's form a size t clique in G.

Suppose that for some $i_1 < i_2$ vertices v_{i_1} and v_{i_2} are not connected by an edge. Consider the set $S \in \mathscr{S}_{i_1,i_2}$ selected in the solution. The size of X is $2^t - 1$ in a standard solution, thus the set X contains at least $2^t - 1 - (2^{i_1} + 2^{i_2})$ elements outside the set S. Therefore, S can be balanced only if all the $2^{i_1} + 2^{i_2}$ elements of S are in X. Assume that the set S selected from $\mathscr{S}_{i_1,i_2}$ is $X_{i_1,u} \cup X_{i_2,v}$. Now $X_{i_1,u} \cup X_{i_2,v} \subseteq X$, which means that $u = v_{i_1}$ and $v = v_{i_2}$. By construction, if $X_{i_1,u} \cup X_{i_2,v}$ is in $\mathscr{S}_{i_1,i_2}$, then u and v are adjacent, hence v_{i_1} and v_{i_2} are indeed neighbors. □

The job of the enforcer systems is to ensure that every solution of weight at most $d = 2^t - 1$ is standard. The $2^t - 1$ blocks $A_{i,j}$ are indexed by two indices i and j. It will be more convenient to index the blocks by a single variable: let $B_1, \ldots, B_{2^t-1}$ be an ordering of the blocks such that B_1 is the only block of group A_0, the blocks B_2, B_3 are the blocks of A_1, the next four blocks are the blocks of A_2, etc.

A naive way of constructing the enforcer set systems would be to have a set system $\mathscr{S}_i$ for each block B_i such that for each element of B_i, there is a corresponding one-element set in $\mathscr{S}_i$ with weight $2^t - 2$. This ensures that if a solution contains at least one element from every block other than B_i, then it has to contain an element of B_i as well. The problem is that every set of $\mathscr{S}_i$ is balanced by the solution $X = \emptyset$, hence such systems cannot ensure that every solution is standard.

There are $2^{2^t-1} - 1$ enforcer set systems: there is a set system $\mathscr{S}_F$ corresponding to each nonempty subset F of $\{1, 2, \ldots, 2^t - 1\}$. The job of $\mathscr{S}_F$ is to rule out the possibility that a solution X contains no elements from the blocks indexed by F, but X contains at least one element from every other block. Clearly, these systems will ensure that no block is empty in a solution, hence every solutions of weight $2^t - 1$ is standard. One possible way of constructing the system $\mathscr{S}_F$ is to have one set of size $|F|$ and weight $2^t - 1 - |F|$ for each possible way of selecting one element from each block indexed by F. Now the problem is that the size of $\mathscr{S}_F$ can be too large, in particular when $F = \{1, 2, \ldots, 2^t - 1\}$. We use a somewhat more complicated construction to keep the size of the systems small.

Given a finite set F of positive integers, define $\mathrm{up}(F)$ to be the largest $\lceil (|F|+1)/2 \rceil$ elements of this set. The enforcer system corresponding to F is defined as

$$\mathscr{S}_F = \prod_{p \in \mathrm{up}(F)} B_p. \tag{3}$$

That is, we consider the blocks indexed by the upper half of F, and put into $\mathscr{S}_F$ all the possible combinations of selecting one element from each block. Let the weight of each set in $\mathscr{S}_F$ be $2^t - 1 - |\mathrm{up}(F)|$. Notice that it is possible that $\mathrm{up}(F_1) = \mathrm{up}(F_2)$ for some $F_1 \neq F_2$, which means that for such F_1 and F_2 the systems $\mathscr{S}_{F_1}$ and $\mathscr{S}_{F_2}$ are in fact the same. However, we do not care about that.

We have to verify that these set systems are not too large, they can be constructed in uniformly polynomial time:

Proposition 5.3. *For every nonempty $F \subseteq \{1, 2, \ldots, 2^t - 1\}$, the enforcer system $\mathscr{S}_F$ contains at most n^2 sets.*

Proof. Let x be the smallest element of $\mathrm{up}(F)$, assume that $2^p \leq x < 2^{p+1}$ for some integer p. There is one block with size n, there are 2 blocks with size $n^{1/2}$, ..., there are 2^i blocks with size $n^{1/2^i}$, hence the size of B_{2^p} is $n^{1/2^p}$. The size of the blocks are decreasing, thus all the sets in the product (3) are of size at most $n^{1/2^p}$. If the smallest element of $\mathrm{up}(F)$ is x, then it can contain at most $x+1$ elements. This means that we take the direct product of at most $x+1$ sets of size at most $n^{1/2^p}$ each. Therefore, the total number of sets in $\mathscr{S}_F$ is at most $(n^{1/2^p})^{x+1} \leq (n^{1/2^p})^{2^{p+1}} = n^2$. □

The following proposition completes the proof of the first direction: if the solution is standard, then we can select a set from each enforcer system. Together with Prop. 5.2, it follows that if there is a clique of size t, then there is a (standard) solution for the constructed instance of CLOSEST SUBSTRING.

Proposition 5.4. *If X is a standard solution, then each $\mathscr{S}_F$ contains a set that is balanced by X.*

Proof. For the enforcer system $\mathscr{S}_F$, let us select the set

$$S_F = X \cap \bigcup_{p \in \mathrm{up}(F)} B_p.$$

That is, S_F contains those vertices of X that belong to the blocks indexed by $\mathrm{up}(F)$. The set S_F is a size $|\mathrm{up}(F)|$ subset of X. Therefore, $|X \triangle S_F| = 2^t - 1 - |\mathrm{up}(F)|$, which is exactly the weight of the selected set. Thus S_F is balanced. □

On the other hand, if there is a solution for the constructed instance of SET BALANCING with $|X| \leq d = 2^t - 1$, then this solution has to be standard, and by Prop. 5.2 there is a clique of size t in G. This completes the proof of the second direction.

Proposition 5.5. *If $|X| \leq 2^t - 1$, then $|X \cap B_i| = 1$ for every block B_i.*

Proof. Assume first that X does not contain elements from some of the blocks. Let F contain the indices of those blocks that are disjoint from X. This means that X contains at least one element from each block not in F, hence $|X| \geq 2^t - 1 - |F|$. Assume that some set S is selected from $\mathscr{S}_F$ in the solution. This set contains elements only from blocks indexed by $\mathrm{up}(F) \subseteq F$, hence S is disjoint from X. Thus $|X \triangle S| = |X| + |S| \geq 2^t - 1 - |F| + |\mathrm{up}(F)| > 2^t - 1 - |\mathrm{up}(F)|$, which means that S is not balanced (here we used $|F| - |\mathrm{up}(F)| < |\mathrm{up}(F)|$). Therefore, each block contains at least one element of X. Since there are $2^t - 1$ blocks, this is only possible if each block contains exactly one element of X. □

The distance $d = 2^t - 1$ and the number $m = \binom{t}{2} + 2^{2^t - 1} - 1$ of the constructed set systems are functions of t only. Each set in the constructed systems has size at most $\ell := 2^t - 1$. The size of each set system is polynomial in n, thus the reduction is a correct parameterized reduction. □

6 Hardness of CLOSEST SUBSTRING

In this section we show that CLOSEST SUBSTRING is W[1]-hard with combined parameters k and d. The reduction is very similar to the reduction presented in [7]. As in that reduction, the main technical trick is that the string s_i is divided into blocks and we ensure that the string s'_i in every solution is one of these blocks.

Theorem 6.1. CLOSEST SUBSTRING *is* W[1]*-hard with parameters d and k, even if $\Sigma = \{0, 1\}$.*

Proof. The reduction is from the SET BALANCING problem, whose W[1]-hardness was shown in Section 5. Assume that m set systems $\mathscr{S}_i = \{S_{i,1}, \ldots, S_{i,|\mathscr{S}_i|}\}$ and an integer d are given. Let $0 \leq w_{i,j} \leq d + \ell$ be the weight of $S_{i,j}$ in $\mathscr{S}_i$, and assume that each set has size at most ℓ. We construct an instance of CLOSEST SUBSTRING where $d+1$ strings $s_{i,1}$, $s_{i,2}$, ..., $s_{i,d+1}$ correspond to each set system $\mathscr{S}_i$, and there is one additional string s_0 called the *template string*. Thus there are $k := (d+1)m + 1$ strings in total.

Set $d' := d + \ell$ and $L := 6d' + 3d'(3d'+1) + |A| + d' - d + 2d'm(d+1)$, where A is the common ground set of the set systems. The template string s_0 has length L, hence $s'_0 = s_0$ in every solution. The string $s_{i,j}$ is the concatenation of *blocks* $B_{i,j,1}$, ..., $B_{i,j,|\mathscr{S}_i|}$ of the same length L, each block corresponds to a set in $\mathscr{S}_i$. We will ensure that in a solution the substring $s'_{i,j}$ is one complete block from $s_{i,j}$. Therefore, selecting $s'_{i,j}$ from $s_{i,j}$ in the constructed CLOSEST SUBSTRING instance plays the same role as selecting a set S_i from $\mathscr{S}_i$ in SET BALANCING.

Each block of the string $S_{i,j}$ is the concatenation of four parts: the front tag, the core, the complete tag, and the back tag. The *front tag* is the same in every block: $1^{3d'}(10^{3d'})^{3d'}1^{3d'}$. The *core* corresponds to the ground set A in the SET BALANCING problem. The length of the core is $|A|$, and the p-th character of the core in block $B_{i,j,k}$ is 1 if and only if the set $S_{i,k} \in \mathscr{S}_i$ contains the p-th element of A. The *complete tag* is $1^{d'-d}$ in every block. The *back tag* is the concatenation of $m(d+1)$ segments $C_{i,j}$ ($1 \leq i \leq m$, $1 \leq j \leq d+1$) (the order in which these segments are concatenated will not be important). The length of each segment is $2d'$. In block $B_{i,j,k}$ of string $s_{i,j}$ the back tag contains 1's only in segment $C_{i,j}$: there is 1 on the first $d' - w_{i,k} \geq 0$ positions of $C_{i,j}$, the rest of $C_{i,j}$ is 0. This completes the description of the strings $s_{i,j}$. Notice that the blocks $B_{i,j_1,k}$ and $B_{i,j_2,k}$ differ only in the back tag. The length L template string s_0 is similar to the blocks defined above: it has the same front tag as all the other blocks, but its core, complete tag, and back tag contains only 0's.

The first direction of the proof is shown in the following proposition:

Proposition 6.2. *If the* SET BALANCING *instance has a solution, then the constructed instance of* CLOSEST SUBSTRING *also has a solution.*

Proof. Let $X \subseteq A$ and $S_{1,a_1} \in \mathscr{S}_1, \ldots, S_{m,a_m} \in \mathscr{S}_m$ be a solution of SET BALANCING. Let the center string s be the concatenation of the front tag, the incidence vector of X, the string $1^{d'-d}$, and the string $0^{2d'm(d+1)}$. The distance of s and s_0 is $|X| + d' - d \leq d'$: the distance is $|X|$ on the core and $d' - d$ on the complete tag. Furthermore, we claim that the block B_{i,j,a_i} in string $s_{i,j}$ is at distance at most d' from s. If we can show this, then it follows that CLOSEST SUBSTRING has a solution.

The front tag of B_{i,j,a_i} is the same as the front tag of s. In the core the distance is the symmetric difference of X and S_{i,a_i}. The complete tag is the same in s and B_{i,j,a_i}. The back tag of s is all 0, while the back tag of B_{i,j,a_i} contains $d' - w_{i,k}$ characters 1 (in the segment $C_{i,j}$). Therefore,

$$d(s, B_{i,j,a_i}) = |X \triangle S_{i,a_i}| + d' - w_{i,k} \leq d',$$

where the inequality follows from the fact that X balances the set S_{i,a_i}, that is, $|X \triangle S_{i,a_i}| \leq w_{i,k}$. □

To prove the reverse direction, we have to show that a solution for the CLOSEST SUBSTRING problem implies a solution of SET BALANCING. We show that it can be assumed that the solution for CLOSEST SUBSTRING is "nice," as defined by the following four propositions (proofs omitted):

Proposition 6.3. *In every solution, the substring $s'_{i,j}$ of $s_{i,j}$ is a block $B_{i,j,b}$ for some value b.*

Proposition 6.4. *If there is a solution for the constructed instance of* CLOSEST SUBSTRING, *then there is such a solution where the front tag of the center string s is the same as the front tag of s_0.*

Proposition 6.5. *If there is a solution for the constructed instance of* CLOSEST SUBSTRING, *then there is a solution where the back tag of the center string s contains only 0's.*

Proposition 6.6. *If there is a solution for the constructed instance of* CLOSEST SUBSTRING, *then there is such a solution where the complete tag of the center string s contains only 1's.*

Assuming that s is of this form, it is not difficult to prove the converse of Prop. 6.2:

Proposition 6.7. *If there is a solution for the constructed instance of* CLOSEST SUBSTRING, *then there is a solution for the* SET BALANCING *problem.*

Proof. Consider a solution where the complete tag of s contains only 1's, and the back tag of s contains only 0's. Define the set $X \subseteq A$ based on the core of s: let an element of A be in X if and only if the corresponding character is 1 in the core of s. The string s differs from the template string s_0 at $|X|$ positions in the core and at $d' - d$ positions in the complete tag. Since $d(s, s_0) \leq d'$, it follows that $|X| \leq d$.

We claim that for every $1 \leq i \leq s$, a set S_i can be selected from $\mathscr{S}_i$ that is balanced by X. Assume that $s'_{i,1}$ is the block $B_{i,1,t}$ for some t. We show that $S_{i,t} \in \mathscr{S}_i$ is balanced by X. Let us determine the distance $d(B_{i,1,t}, s)$, which is by assumption at most d'. In the core, the two strings differ on the symmetric difference of $S_{i,t}$ and X. The strings do not differ on the complete tag, but they differ on every position of the back tag where $B_{i,1,t}$ is 1. There are exactly $d' - w_{i,t}$ such positions, hence

$$d(s, s'_{i,j}) = |X \triangle S_{i,k}| + d' - w_{i,t} \leq d',$$

which means that $|X \triangle S_{i,t}| \leq w_{i,t}$ and $S_{i,t}$ is balanced. □

Prop. 6.2 and 6.7 together prove the correctness of the reduction. □

Putting together Theorem 5.1 and 6.1 gives a two-step reduction from MAXIMUM CLIQUE to CLOSEST SUBSTRING. Given an instance of MAXIMUM CLIQUE with parameter t, this two-step reduction constructs an instance of CLOSEST SUBSTRING with parameters $d = 2^{O(t)}$ and $k = 2^{2^{O(t)}}$. It is unlikely that MAXIMUM CLIQUE can be solved in $f(t) \cdot n^{o(t)}$ time: that would imply that 3-SAT could be solved in subexponential time [3]. Using our reduction, we can transfer this lower bound on MAXIMUM CLIQUE to the CLOSEST SUBSTRING problem:

Corollary 6.8. *There is no $f_1(k,d) \cdot n^{o(\log d)}$ or $f_2(k,d) \cdot n^{o(\log\log k)}$ time algorithm for* CLOSEST SUBSTRING, *unless* 3-SAT *can be solved in subexponential time.* □

Cesati and Trevisan [2] have shown (by an easy argument) that if a problem is W[1]-hard, then the corresponding optimization problem cannot have an EPTAS (i.e., a PTAS with running time $f(\varepsilon) \cdot n^c$), unless FPT = W[1]. Hence the $n^{O(1/\varepsilon^4)}$ time PTAS of [13] for CLOSEST SUBSTRING cannot be improved to an EPTAS. Furthermore, the connection with subexponential algorithms allows us to give a lower bound on the exponent of n:

Corollary 6.9. *There is no $f(\varepsilon) \cdot n^{o(\log 1/\varepsilon)}$ time PTAS for* CLOSEST SUBSTRING, *unless* 3-SAT *can be solved in subexponential time.* □

7 Conclusions

We have proved that the CLOSEST SUBSTRING problem parameterized by the distance parameter d and by the number of strings k is W[1]-hard, even if the alphabet is binary. This improves the previous result of [7], where it is proved that the problem is W[1]-hard with parameter k only (and binary alphabet). Our hardness result also improves [11], where it is proved that DISTINGUISHING SUBSTRING SELECTION (a generalization of CLOSEST SUBSTRING) is W[1]-hard with parameters k and d (again with binary alphabet). In our reduction we used some of the techniques from these results, but new ideas were also required.

The W[1]-hardness of a problem is usually interpreted as evidence that the problem is unlikely to be fixed-parameter tractable, that is, the parameter has to appear in the exponent of n. Furthermore, using recent connections with subexponential algorithms, we can even give a lower bound on the exponent of n. Our reduction is "weak" in the sense that the parameters are significantly increased (exponentially and double exponentially). Therefore, we obtain only weak lower bounds on the exponent of n: all we can show is that the exponent cannot be $o(\log d)$ or $o(\log\log k)$. However, it turned out that these bounds are tight: we presented two algorithms where the exponent of n is $O(\log d)$ and $O(\log\log k)$, respectively. The second algorithm is based on some surprising connections with the extremal combinatorics of hypergraphs. We have introduced and investigated the half-covering property, which played an important role in the algorithm. Furthermore, we have shown that all the copies of hypergraph H in hypergraph G can be efficiently found if H has small fractional cover number. This result might be useful in some other applications as well.

Our results present an example where parameterized complexity and subexponential algorithms are closely connected. First, a weak parameterized reduction might be the sign that some kind of subexponential algorithm is possible for the problem. On the other hand, a parameterized reduction can be used to show the optimality of a subexponential algorithm. It is possible that this interplay between parameterized complexity and subexponential algorithms appears in the case of some other problems as well.

The W[1]-hardness of CLOSEST SUBSTRING implies that there is no EPTAS for the optimization version of the problem. More precisely, we can show that there is no PTAS with running time $f(\varepsilon) \cdot n^{o(\log 1/\varepsilon)}$, unless there are subexponential algorithms for 3-SAT. However, it does not rule out the possibility that the $n^{O(1/\varepsilon^4)}$ time PTAS of [13] can be improved to $n^{O(\log 1/\varepsilon)}$. It is an intriguing open question to determine whether such an improvement is possible.

References

[1] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *J. Comput. System Sci.*, 18(2):155–193, 1979.

[2] M. Cesati and L. Trevisan. On the efficiency of polynomial time approximation schemes. *Inform. Process. Lett.*, 64(4):165–171, 1997.

[3] J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. In *Proceedings of 19th Annual IEEE Conference on Computational Complexity*, pages 150–160, 2004.

[4] F. R. K. Chung, R. L. Graham, P. Frankl, and J. B. Shearer. Some intersection theorems for ordered sets and graphs. *J. Combin. Theory Ser. A*, 43(1):23–37, 1986.

[5] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer-Verlag, New York, 1999.

[6] P. A. Evans, A. D. Smith, and H. T. Wareham. On the complexity of finding common approximate substrings. *Theoret. Comput. Sci.*, 306(1-3):407–430, 2003.

[7] M. R. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of motif search problems. To appear in Combinatorica.

[8] M. R. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of Closest Substring and related problems. In *STACS 2002*, volume 2285 of *Lecture Notes in Comput. Sci.*, pages 262–273. Springer, Berlin, 2002.

[9] M. Frances and A. Litman. On covering problems of codes. *Theory Comput. Syst.*, 30(2):113–119, 1997.

[10] E. Friedgut and J. Kahn. On the number of copies of one hypergraph in another. *Israel J. Math.*, 105:251–256, 1998.

[11] J. Gramm, J. Guo, and R. Niedermeier. On exact and approximation algorithms for distinguishing substring selection. In *Fundamentals of computation theory*, volume 2751 of *Lecture Notes in Comput. Sci.*, pages 195–209. Springer, Berlin, 2003.

[12] J. Gramm, R. Niedermeier, and P. Rossmanith. Fixed-parameter algorithms for closest string and related problems. *Algorithmica*, 37(1):25–42, 2003.

[13] M. Li, B. Ma, and L. Wang. On the closest string and substring problems. *J. ACM*, 49(2):157–171, 2002.