Boucher and Ma *BMC Bioinformatics* 2011, **12**(Suppl 1):S55
http://www.biomedcentral.com/1471-2105/12/S1/S55

BMC Bioinformatics

RESEARCH Open Access

CLOSEST STRING WITH OUTLIERS

Christina Boucher*, Bin Ma*

From The Ninth Asia Pacific Bioinformatics Conference (APBC 2011)
Inchon, Korea. 11-14 January 2011

Abstract

Background: Given n strings $s_1, \ldots, s_n$ each of length ℓ and a nonnegative integer d, the CLOSEST STRING problem asks to find a center string s such that none of the input strings has Hamming distance greater than d from s. Finding a common pattern in many – but not necessarily all – input strings is an important task that plays a role in many applications in bioinformatics.

Results: Although the closest string model is robust to the oversampling of strings in the input, it is severely affected by the existence of outliers. We propose a refined model, the CLOSEST STRING WITH OUTLIERS (CSWO) problem, to overcome this limitation. This new model asks for a center string s that is within Hamming distance d to at least $n - k$ of the n input strings, where k is a parameter describing the maximum number of outliers.
A CSWO solution not only provides the center string as a representative for the set of strings but also reveals the outliers of the set.
We provide fixed parameter algorithms for CSWO when d and k are parameters, for both bounded and unbounded alphabets. We also show that when the alphabet is unbounded the problem is W[1]-hard with respect to $n - k$, ℓ, and d.

Conclusions: Our refined model abstractly models finding common patterns in several but not all input strings. We initialize the study of the computability of this model and show that it is sensitive to different parameterizations. Lastly, we conclude by suggesting several open problems which warrant further investigation.

Background

Finding similar regions in multiple DNA, RNA, or protein sequences plays an important role in many applications, including universal PCR primer design [1-4], genetic probe design [2], antisense drug design [2,5], finding transcription factor binding sites in genomic data [6], determining an unbiased consensus of a protein family [7], and motif-recognition [2,8,9]. The CLOSEST STRING problem formalizes these tasks and can be defined as follows: given a set of n strings S of length ℓ over the alphabet Σ and parameter d, the aim is to determine if there exists a string s that has Hamming distance at most d from each string in S. The optimization version of this problem tries to minimize the parameter d. We refer to s as the *center string* and let $d(x, y)$ be the Hamming distance between strings x and y.

The CLOSEST STRING was first introduced and studied in the context bioinformatics by Lanctot *et al.* [2]. Frances and Litman [10] showed the problem to be NP-complete even in the special case when the alphabet is binary, implying there is unlikely to be a polynomial-time algorithm for this problem unless P = NP. Since its introduction the investigation of efficient polynomial time approximation algorithms and exact exponential time algorithms for the CLOSEST STRING problem has been thoroughly considered [2,11-16].

The CLOSEST STRING problem requires that the Hamming distance constraint be satisfied for each of the input strings and therefore, is robust to the oversampling of the input strings. For this reason it is frequently used to model many of the aforementioned applications. However, this property also causes a severe problem: if the input includes a string that is

* Correspondence: cabouche@cs.uwaterloo.ca; binma@cs.uwaterloo.ca
David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON

significantly different from the other input strings, which we refer to as an "outlier", then it will have the effect of causing there not to exist a center string for the complete set of input strings; d will have to be increased dramatically to account for this string and obtain a center string. This is a significant limitation for applications such as the design of universal primers where a small d is crucial for the effectiveness of the primers. In this and many other applications, it would be preferable to determine a "good" center string (*i.e.* one that is reasonably close to each of the strings) for a large portion of the input strings rather than trying to find a center string for the complete set and in doing so finding one that is far distance from many or all of the strings. Hence, we aim to model the task of finding a center string that is within distance d to most – but not necessarily all – of the input strings, where d is reasonably small. Another compelling consequence of the modification of the model is that in situations where a more satisfying solution can be found by regarding a few strings as outliers, the initial decision of including them requires reexamination.

We formally model this problem as follows:

CLOSEST STRING WITH OUTLIERS (CSWO)

INPUT: A set of n length-ℓ strings $S = \{s_1,..., s_n\}$ over a finite alphabet Σ and nonnegative integers k and d.

QUESTION: Find a center string s and a subset of $S^* \subset S$, such that $|S^*| = n - k$ and $d(s, t) \leq d$ for $t \in S^*$.

For the rest of the paper we denote $n - k$ as n^*, and $s_i[p]$ to be the symbol at position p of string s_i.

There exists a simple reduction from the CLOSEST STRING problem to CSWO that demonstrates it is NP-complete even in the special case where the alphabet is binary and $k = 0$, implying it is unlikely to be solved exactly by a polynomial-time algorithm, unless P=NP. One approach to investigating the computational intractability of CSWO is to consider its parameterized complexity, which aims to classify computationally hard problems according to their inherent difficulty with respect to multiple parameters of the input. If it is solvable by an algorithm that is polynomial in the input size and exponential in parameters that are typically small then it can still be considered tractable in some practical sense.

For unbounded alphabet size, we show that CSWO is W[1]-hard for every combination of the parameters ℓ, d, and n^* and thus, is fixed parameter intractable when parameterized by any subset of these parameters, unless FPT = W[1]. We also show that when the alphabet is unbounded, there exists a fixed parameter tractable algorithm for CSWO with respect to the parameters d and k. In the case of constant size alphabet, CSWO is fixed parameter tractable for the parameter n but intractable for the parameter k. The complexity of the problem remains open when parameterized by d and the alphabet is of constant size, and when parameterized by n^* and k.

Previous Results

It is worth noting that analogous parameterized complexity studies have been performed for the CLOSEST STRING problem and the CLOSEST SUBSTRING problem. Gramm *et al.* [13] demonstrated that the CLOSEST STRING problem is FPT when the number of strings remains fixed. This FPT result is based on an integer linear programming formulation with a constant number of variables (assuming n is fixed), and the application of the result of Lenstra [19] that proves integer linear programming is polynomial-time solvable when the number of variables remains fixed. They further demonstrated that the problem is FPT when d is a parameter by giving a $O(n\ell + nd(d + 1)^d)$ time algorithm [13]. Ma and Sun gave an $O(n|\Sigma|^{O(d)})$ algorithm, which is a polynomial-time algorithm when $d = O(\log n)$ and Σ has constant size [16]. Chen *et al.* [20], Wang and Zhu [21], and Zhao and Zhang [22] improved upon the fixed parameter tractable result of Ma and Sun [16].

The CLOSEST SUBSTRING problem seems to be inherently more intractable then the CLOSEST STRING problem. Given n strings $s_1, s_2,..., s_n$ over alphabet Σ and integers d and ℓ, the CLOSEST SUBSTRING problem aims to determine whether there is a string s of length ℓ such that, for all $i = 1,..., n$, $d(s, s_i') \leq d$ where s_i' is a length ℓ substring of s_i. Fellows *et al.* [11] showed that CLOSEST SUBSTRING is W[1]-hard with respect to the number of input strings n even for a binary alphabet. When Σ is unbounded the problem is W[1]-hard with respect to the parameters ℓ, d and n[11]. Most recently, Marx [23] proved the problem is W[1]-hard with combined parameters n and d even if the alphabet is binary, which resolved an open problem stated in [11,12,24].

Methods

We give insight into the computational tractability of CSWO through studying the parameterized complexity of the problem. Parameterized complexity aims to classify problems according to their inherent difficulty with respect to multiple parameters of the input.

Parameterized Complexity

A problem φ is said to be *fixed parameter tractable* with respect to parameter k if there exists an algorithm that solves φ in $f(k) \cdot n^{O(1)}$ time, where f is a function of k that is independent of n [17]. Given a graph $G = (V, E)$ with vertex set V, edge set E, and positive integer k, the Vertex Cover problem aims to discern where there is a subset of vertices $V_c \subseteq V$ with k or fewer vertices such

that each edge in E has at least one its endpoints in V_c. The vertex cover problem is NP-complete [18] but is fixed parameter tractable since there exists algorithmic solutions that have running time $O(kn + 1.3^k)$ [17]. The corresponding complexity class is FPT.

Not all NP-complete problems are in FPT. For example, consider the NP-complete CLIQUE problem: given an undirected graph $G = (V, E)$ and a positive integer t, the aim is determine whether there is a subset of vertices $C \subseteq V$ of size at least t where each pair of vertices in C are connected by an edge. The best known algorithms for solving clique runs in time $O(n^{o(t)})$ and hence, there is no known algorithm for solving t for which t is not in the exponent of n in the running time [17].

In order to characterize those problems that do not seem to admit a fixed parameter efficient algorithm, Downey and Fellows [17] defined a *fixed parameter reduction*. We will restrict interest to the W[1] class and hence, the following definition will only apply to W[1]-hardness. W[1]-hardness gives convincing evidence that a parameterized problem with parameter k is unlikely to have an algorithm that has running time of the form $f(k) \cdot n^{O(1)}$. Let $L, L' \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized languages, then L *reduces* to L' if there are functions $k \to k'$ and $k \to k''$ from $\mathbb{N}$ to $\mathbb{N}$ and a function $(x, k) \to x'$ from $\Sigma^* \times \mathbb{N}$ to Σ^* such that:

1. $(x, k) \to x'$ is computable in time $k''|x|^c$, for some constant c and
2. $(x, k) \in L$ if and only if $(x', k') \in L'$.

Results and Discussion

In the following subsections, we study the parameterized tractability of CSWO and show the problem is sensitive to different parameterizations.

CSWO: Tractability Results

We first consider when Σ is a parameter. In computational biology applications the biological sequences of interest are typically DNA or protein sequences, hence the number of different symbols is a small constant (*i.e.* 4 or 20 in the case of DNA or protein sequences, respectively). Restricting Σ only does not make CSWO tractable since it is NP-hard even when the alphabet is binary. However, if Σ and ℓ are both parameters then it is fixed-parameter tractable; we can enumerate and check all the $|\Sigma|^\ell$ possible center strings. As a result the problem is fixed parameter tractable with the combined parameters Σ, ℓ, d and n^*. We will prove in a later section that it is imperative that Σ be a parameter in order to obtain this tractability.

Next we show that CSWO is fixed parameter tractable if d and k are parameters. The fixed parameter algorithm that we present is similar to the algorithm presented by Gramm *et al.* [13], where it is proved that CLOSEST STRING is fixed parameter tractable with respect to the parameter d. In the algorithm by Gramm *et al.* [13] at each recursive step a string s is selected that has Hamming distance at least $d + 1$ away from the current candidate center string x if one exists; otherwise x is returned since it is a center string. Then for any $d + 1$ positions where x and s disagree, there is at least one position at which s is equal to the final solution. The algorithm tries each of the $d + 1$ positions, changes x to s at one of the $d + 1$ the position, reduces Δd by one, and calls itself recursively. Hence, Δd is the current degeneracy parameter at a particular recursive iteration and x is the current candidate center string. Since the recursion stops after at most d steps the size of the search tree is bounded by $O((d + 1)^d)$.

CSWO Algorithm

Input: A CSWO instance with a set of S n strings of length ℓ, parameters Δd, d and k, and a candidate string x.

Output: A string s^* if there exists a set S of at least n^* strings where each string in S has distance at most d from s^*, and "Not found" otherwise.

1. If $\Delta d < 0$ or $k < 0$ then return "Not found".
2. Choose $i \in \{1,..., n\}$ such that $d(x, s_i) > d$. If no such i exists return x.
3. s_{ret} = CSWO Algorithm $(S \setminus \{s_i\}, \Delta d, k - 1, x)$.
4. If s_{ret} = "not found " then:
(a) $\boldsymbol{P} = \{p \mid \times [p] \neq s_i[p]\}$;
(b) Choose any $\boldsymbol{P}'$ from $\boldsymbol{P}$ with $|\boldsymbol{P}'| = d + 1$.
(c) For each position $p \in \boldsymbol{P}'$:
- Let x be equal to s_i at position p.
- s_{ret} = CSWO Algorithm $(S, \Delta d - 1, k, x)$.
- If $s_{ret} \neq$ "not found", then return s_{ret}.

5. Return "not found".

Our algorithm begins with s_1 as the candidate center string. If s_1 is a center string with respect to S then we are done; otherwise there exists a string s_i that has distance at least $d + 1$ from s_1. We "guess" whether s_i belongs in the set of outliers. If it is an outlier then we remove it from S and recurse on the smaller set with $k - 1$. If it is not an outlier then we use s_i to move the candidate string x closer to toward s_i, which can be done by applying the methodology of Gramm *et al.* [13]. We use the term "guess" as an euphemism in this brief description of the our algorithm but rather we try both possibilities as can be seen in the CSWO Algorithm. This will increase the size of the search tree.

Proposition 1 *The CSWO Algorithm solves the CSWO problem in time* $O(n\ell + nd \cdot d^d \cdot 2^{k+d})$.

Proof. Running time. Each recursion of the algorithm reduces either k or d by 1. Thus, there are at most $k + d$ guesses of whether a particular string belongs in the set of outliers. Thus, the search tree size is increased by a multiplicative factor of at most 2^{k+d} and the search tree

size is bounded above by $O(2^{k+d} \cdot (d+1)^d)$. The analysis of Gramm *et al.* [13] demonstrated that each recursive step takes time $O(nd)$ and the preprocessing time takes $O(n\ell)$ and therefore, we obtain an overall running time of $O(n\ell + nd \cdot d^d \cdot 2^{k+d})$.

Correctness We show the correctness of the algorithm by showing the correctness of the first recursive step and then the correctness of the algorithm follows by inductively applying the following argument. Clearly, if S does not contain a subset S^* of n^* strings, such that there exists a center string s^* for S^* then "not found" will be returned and therefore, we assume otherwise.

If s_1 is a center string for S then the algorithm immediately halts so we assume there exists a string s_i in S that does not have s_1 as a center string. CSWO Algorithm creates two subcases: one where s_i is in the set of outliers, and another where s_i is not. Suppose s_i is in the set of outliers then the first case will successfully remove s_i from the set and recurse on $S \backslash \{s_i\}$. Otherwise, if s_i is not in the set of outliers then eventually the second case will reached. We refer to the set of positions as *correct* if $\{p \mid s_1[p] \neq s^*[p] = s[p]\}$. It follows from Gramm *et al.* [13] that one of the $d+1$ chosen positions p will be a correct one. Thus, we have shown that either one of the subcases will lead to a smaller subcase containing the solution for S.

The previous result demonstrates the fixed parameter tractability with respect to d and k. We note that a similar modification of the $O(n|\Sigma|^{O(d)})$ algorithm of Ma and Sun [16] also gives a fixed parameter algorithm with respect to the parameters Σ, d and k. In the modified algorithm, for any string s with distance greater than d to the current candidate center string x, we again try the subcases where s is an outlier, and is not an outlier. In the former case, we remove s from the set of input strings S and recurse on S and $k-1$, and in the latter case, we use the same technique as in the algorithm of Ma and Sun [16] to reduce the distance between x and the final solution. This modification that accounts for the outliers results an extra multiplicative factor of $O(2^{k+\log d})$ to the running time of the original algorithm. Although this algorithm improves upon the running time of the previous result, it requires that Σ is also a parameter. Further, we note that some of the recent improvements [20-22] to the algorithm of Ma and Sun can be modified in a similar manner to obtain fixed parameter algorithms for CSWO with respect to parameters Σ, d and k.

Proposition 2 CSWO *is fixed parameter tractable for parameters* Σ *and* n.

Proof. Gramm *et al.* [13] gave a linear fixed parameter tractable algorithm for CLOSEST STRING with respect to the number of strings and Σ, which we refer to this algorithm as *ILP-procedure(S)*, where S is the set of input strings. Our algorithm enumerates all size-n^* subsets of S, and call *ILP-procedure* on each subset.

CSWO: Intractability Results

We derive the W[1]-hardness result by a series of intermediate steps, aiming at a reduction from Clique to CSWO, showing that CSWO is W[1]-hard for the combination of ℓ, d, and n^*, and when the alphabet is unbounded.

Reduction from CLIQUE

As previously described, we let the CLIQUE instance be given by an undirected graph $G = (V, E)$ with a set $V = \{v_1, v_2, \ldots, v_n\}$ of n vertices, a set E of m edges, and a positive integer t denoting the size of the desired clique. We describe how to generate a set S of $\binom{t}{2}|E|$ strings such that G has a clique of size t if and only if there is a subset of S of size $\binom{t}{2}$, denoted as S^*, where there exists a string x such that $d(s_i, x) \leq d$ for all $s_i \in S^*$. We let $\ell = t$ and $d = t - 2$. We assume that $t > 2$ since $t \leq 1$ produces trivial cases.

We begin by describing the alphabet. We assume $|\Sigma|$ can be infinite and we let Σ be equal to the union of the following sets of symbols:

1. $\{v_i |$ for all $i = 1, \ldots, |V|\}$. Hence, there exists one symbol representing each vertex in G.
2. $\{c_{i,j,m} | i = 1, \ldots, t;\ j = 1, \ldots, t;\ m = 1, \ldots, |E|\}$. There exists an unique symbol for each $\binom{t}{2} \cdot |E|$ strings produced for our reduction.

Hence, we have a total of $|V| + \binom{t}{2} \cdot |E|$ number of symbols.

Next, we generate a set of $\binom{t}{2}|E|$ strings $S = \{s_{1,1,1}, \ldots, s_{1,1,|E|}, s_{1,2,1}, \ldots, s_{1,2,|E|}, \ldots, s_{t-1,t,|E|}\}$. Every string has length t and will encode one edge of the input graph. There will be $\binom{t}{2}$ corresponding for each edge, however, encode the edges in different positions. For string $s_{i,j,m}$ we encode edge $e_m = (v_r, v_s)$, where $1 \leq r < s \leq |V|$, but letting position i equal to v_r and position j equal to v_s and the remaining positions equal to $c_{i,j,m}$. Hence, a string is given by

$$s_{i,j,m} := [c_{i,j,m}]^{i-1}\, v_r [c_{i,j,m}]^{j-i-1}\, v_s [c_{i,j,m}]^{m-j}.$$

To clarify our reduction, we give an example. Let $G = (V, E)$ be an undirected graph with $V = v_1, v_2, v_3, v_4$ and edges $E = \{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_2, v_3)\}$ and let our CLIQUE instance have G and $t = 3$. Figure 1 illustrates

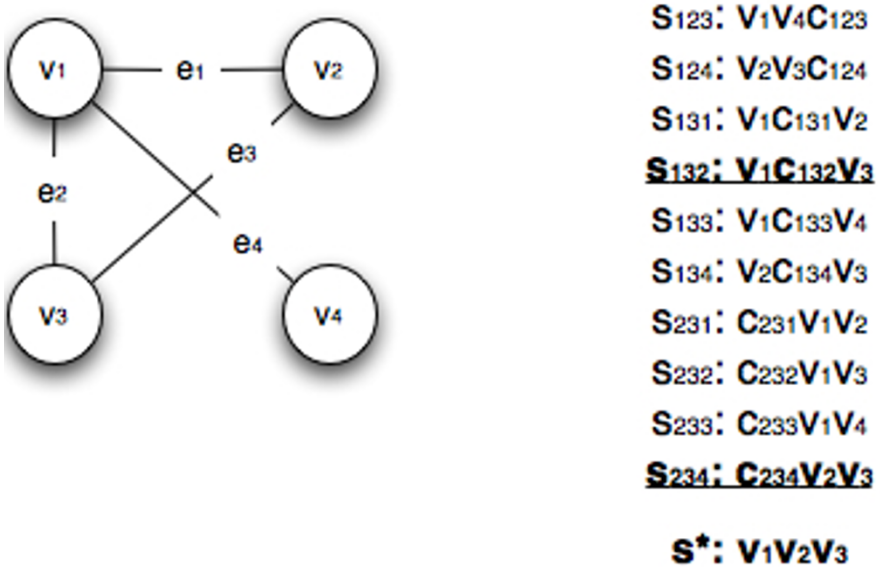

Figure 1 An illustration of the reduction from CLIQUE to CSWO Example of the reduction from a CLIQUE instance G with $t = 3$ to an instance of CSWO with 12 strings with $\ell = t = 3$, $d = t - 2 = 1$, and $\binom{t}{2}$. In bold we have the set of strings $S^* = \{s_{121}, s_{132}, s_{234}\}$ that corresponds to the clique containing the vertices $\{v_1,v_2,v_3\}$. We note that S^* is the only set such that $|S^*| = 3$, which have Hamming distance at most d from s^*.

the reduction. Using G, we exhibit the above construction of $\binom{t}{2} \cdot |E| = 12$ strings, which we denote as S. We claim that there exists a clique of size 3 if and only if there exists a string s^* of length $\ell = t = 3$ and subset S^* of S of size 3 where $d(s,s_i) \leq d$ for all $s_i \in S^*$. In this example the center string s is equal to $v_1v_2v_3$ and each string in the set $\{v_1v_2c_{121}, v_1c_{132}v_3, c_{234}v_2v_3\}$ is such that each string in S^* has Hamming distance at most 1 from s.

Correctness of the Reduction

The following two lemmas establish the correctness of the reduction.

Lemma 1 For a graph with a t-clique, the construction in Subsection produces a CSWO instance with a set S* and a string s of length such that for every $s_i \in$ S* $d(s_i, s) \leq d$.

Proof Let the input graph have a clique of size t. Let $v_{\alpha 1}, v_{\alpha 2}, \ldots, v_{\alpha t}$ be the vertices in the clique C of size t and without loss of generality, assume $\alpha_1 < \alpha_2 < \ldots < \alpha_t$. Then we claim that the there exists a subset of $\binom{t}{2}$ vertices that have distance at most $t - 2$ from the string $s = v_{\alpha 1}v_{\alpha 2} \ldots v_{\alpha t}$. Consider the first edge of the clique $(v_{\alpha 1}, v_{\alpha 2})$ of the clique then it follows that the string $s_{11r} = v_{\alpha 1}v_{\alpha 2}\,[c_{11r}]^{t-2}$, where edge r has endpoints $v_{\alpha 1}v_{\alpha 2}$, is contained in the set of strings $\{s_{111}, s_{112}, \ldots, s_{11|E|}\}$. Clearly, $H(s_{11r},s) = t - 2$. For each edge in C we have we have a string in S that has distance at most $t - 2$ from s and our lemma follows from this construction.

For the reverse direction, we need to prove that the existence a subset S^* of $\binom{t}{2}$ and a string s where $d(s, s_i) \leq t - 2$ for all $s_i \in S^*$ implies the existence of a clique in G with t vertices.

Lemma 2 *The t symbols of the center string correspond to the t vertices of clique in the input graph*

Proof. Let S^* be the subset of S of size $n^* = \binom{3}{2} = 3$ such that s has distance $t - 2$ from each string in S^*. Since $\ell = t$, $n^* = t$, $d = t - 2$ and for each symbol $c_{i,j,m}$ there exists only a single string $i = 1, \ldots,t$, $j = 1, \ldots,t$ and $m = 1,\ldots, |E|$ it follows from the Pigeonhole principle that the center string s only contains symbols from $\{v_i|$ for all $i = 1,\ldots,|V|\}$. Without loss of generality assume s

Table 1 Parameterized tractability of CSWO

Parameter(s)	$\|\Sigma\|$ is a parameter	$\|\Sigma\|$ is unbounded
ℓ, d, n^*	FPT (trivial)	W[1]-hard (*)
ℓ	FPT (trivial)	W[1]-hard (*)
d, n^*	Open	W[1]-hard (*)
d, k	FPT (*)	FPT (*)
n^*, k	FPT	Open
k	W[1]-hard (trivial)	W[1]-hard (trivial)

An overview of the fixed parameter tractability and intractability of the CSWO. Asterisk denotes the new results found in this paper.

is equal to $v_{\alpha 1}v_{\alpha 2}$ ...$v_{\alpha t}$ for α_{v1}, α_{v2},...,$\alpha_{vt} \in \{1,..., |V|\}$. Consider any pair α_i, α_j for $1 \le i < j \le t$ and consider the set of strings $S_{i,j} = \{s_{i,j,1}, s_{i,j,2},..., s_{i,j,|E|}\}$. Recall that $S_{i,j}$ contains a string corresponding to each edge $e = (r, s)$ in E which has v_r at the ith position and v_s at the jth position and $c_{i,j,m}$ at all remaining positions. Therefore, we can only find a string in $S_{i,j}$ that has distance at most $t - 2$ from s if v_{α_i} is at the ith position and $v_{\alpha j}$ is at the jth position; and such a string exists if and only if there is an edge in G connecting $v_{\alpha i}$ to $v_{\alpha j}$. Hence, the center string s implies there exists an edge between any pair of vertices in G in the set $\{v_{\alpha 1}v_{\alpha 2} \ldots v_{\alpha t}\}$ and by definition the vertices form a clique.

Our main theorem follows directly from Lemma 1 and Lemma 2. We note that the hardness for the combination of all three parameters also implies the hardness for each subset of the three.

Theorem 1 CSWO *with unbounded alphabet is W[1]-hard with respect to the parameters* ℓ, d, *and* n^*.

Since there exists a trivial reduction from the CLOSEST STRING problem to CSWO (*i.e.* simply set $k = 0$ in CSWO), there cannot exist a fixed parameter tractable algorithm for CSWO with k as a parameter, unless P = NP; such an algorithm would contradict the NP-hardness of CLOSEST STRING.

Fact 1 CSWO *is W[1]-hard with respect to the parameter k and when* $|\Sigma| \le 2$, *unless P = NP.*

Conclusions

We introduced the CSWO problem, and proved with unbounded alphabet size and parameterized by ℓ, d and n^* it is W[1]-hard. We also gave fixed parameter algorithms for the problem when parameterized by d and k, and with unbounded alphabet size. In the case of a fixed alphabet size, we showed CSWO is fixed parameter tractable when parameterized by $n = n^* + k$. Table 1 summarizes these tractability and intractability results.

Currently, the fixed parameter tractability of the CSWO problem when parameterized by d, n^* and Σ, and by n^* and k, remains open (see Table 1). In addition, the existence of efficient, non-trivial approximation algorithms for this problem warrants further investigation.

Acknowledgement

CB is supported by NSERC Grant OGP0046506, NSERC Grant OGP0048487, Canada Research Chair program, MITACS, and Premier's Discovery Award. BM is supported by NSERC (RGPIN 238748-2006), China 863 National High-tech R&D Program (2008AA02Z313), and a startup grant at University of Waterloo. We are also grateful to the referees for their many helpful comments.

This article has been published as part of *BMC Bioinformatics* Volume 12 Supplement 1, 2011: Selected articles from the Ninth Asia Pacific Bioinformatics Conference (APBC 2011). The full contents of the supplement are available online at http://www.biomedcentral.com/1471-2105/12?issue=S1.

Authors' contributions

Concept, FPT analysis: BM. W[1]-hardness analysis: CB. Manuscript preparation: BM and CB.

Competing interests

The authors declare that they have no competing interests.

Published: 15 February 2011

References

1. Dopazo J, Rodríguez A, Sáiz J, Sobrino F: **Design of primers for PCR amplification of highly variable genomes.** *Computer Applications in the Biosciences* 1993, **9**:123-125.
2. Lanctot J, Li M, Ma B, Wang S, Zhang L: **Distinguishing string selection problems.** *Information and Computation* 2003, 41-55.
3. Lucas K, Busch M, Össinger S: **Thompson J: An improved microcomputer program for finding gene-and gene family-specific oligonucleotides suitable as primers for polymerase chain reactions or as probes.** *Computer Applications in the Biosciences* 1991, **7**:525-529.
4. Proutski V, Holme E: **Primer master: A new program for the design and analyiss of PCR primers.** *Computer Applications in the Biosciences* 1996, **12**:253-255.
5. Deng X, Li G, Li Z, Ma B, Wang L: **Genetic design of drugs without side-effects.** *SIAM Journal on Computing* 2003, **32(4)**:1073-1090.
6. Tompa M, Li N, Bailey TL, Church GM, De Moor B, Eskin E, Favorov AV, Frith MC, Fu Y, Kent WJ, et al.: **Assessing computational tools for the discovery of transcription factor binding sites.** *Nature Biotechnology* 2005, **23**:137-144.
7. Ben-Dor A, Lancia G, Perone J, Ravi R: **Banishing bias from consensus strings.** *Proc. of 8th CPM* 1997, 247-261.
8. Pavesi G, Mauri G, Pesole G: **An algorithm for finding signals of unknown length in DNA sequences.** *Bioinformatics* 2001, **17**:S207-S214.
9. Pevzner P, Sze S: **Combinatorial approaches to finding subtle signals in DNA strings.** *Proc. of 8th ISMB* 2000, 269-278.
10. Frances M, Litman A: **On covering problems of codes.** *Theoretical Computer Science* 1997, **30(2)**:113-119.
11. Fellows M, Gramm J, Neidermeier R: **On the Parameterized Intractability of Closest Substring and Related Problems.** *Proc. of 19th STACS* 2002, 262-273.
12. Fellows M, Gramm J, Niedermeier R: **On The Parameterized Intractability Of Motif Search Problems.** *Combinatorica* 2006, **26**:141-167.
13. Gramm J, Niedermeier R, Rossmanith P: **Fixed-parameter algorithms for closest string and related problems.** *Algorithmica* 2003, **37**:25-42.
14. Li M, Ma B, Wang L: **Finding similar regions in many strings.** *Journal of Computer and System Sciences* 2002, **65**:73-96.
15. Ma B: **A polynomial time approximation scheme for the closest substring problem.** *Proc. of 11th CPM* 2000, 99-107.
16. Ma B, Sun X: **More efficient algorithms for closest string and substring problems.** *Proc. of 12th ACM RECOMB* 2008, 396-409.
17. Downey R, Fellows M: **Parameterized Complexity.** Springer; 1999.
18. Garey M, Johnson D: **Computers and Intractability: A Guide to the Theory of NP-Completeness.** W. H. Freeman; 1979.

19. Lenstra W: **Integer programming with a fixed number of variables.** *Mathematics of Operations Research* 1983, **8**:538-548.
20. Chen ZZ, Ma B, Wang L: **A Three-String Approach to the Closest String problem.** *Proc. of 16th COCOON (to appear)* 2010.
21. Wang L, Zhu B: **Efficient algorithms for the closest string and distinguishing string selection problems.** *Proc. of 3rd FAW* 2009, 261--270.
22. Zhao R, Zhang N: **A more efficient closest string algorithm.** *Prof. of 2nd BICoB (to appear)* 2010.
23. Marx D: **Closest Substring Problems with Small Distances.** *SIAM Journal on Computing* 2008, **38**:1382-1410.
24. Gramm J, Guo J, Niedermeier R: **On Exact and Approximation Algorithms for Distinguishing Substring Selection.** *Proc. of 14th FCT* 2003, 261-272.

doi:10.1186/1471-2105-12-S1-S55
Cite this article as: Boucher and Ma: CLOSEST STRING WITH OUTLIERS. *BMC Bioinformatics* 2011 **12**(Suppl 1):S55.