An efficient algorithm for the Closest String Problem

Omar Vilca[1], **Rosiane de Freitas**[1]

[1]Institute of Computation – Federal University of Amazonas (UFAM)
Manaus, AM – Brasil

{omarlatorre,rosiane}@icomp.ufam.edu.br

Abstract. *The closest string problem that arises in computational molecular biology and coding theory is to find a string that minimizes the maximum Hamming distance from a given set of strings, the CSP is NP-hard problem. This article proposes an efficient algorithm for this problem with three strings. The key idea is to apply normalization for the CSP instance. This enables us to decompose the problem in five different cases corresponding to each position of the strings. Furthermore, an optimal solution can be easily obtained in linear time. A formal proof of the algorithm will be presented, also numerical experiments will show the effectiveness of the proposed algorithm.*

1. Introduction

String selection problems are among the most important topics facing researchers in computational biology. Combinatorial optimization is a possible approach to solving selection sequences problems. There are some previously exact algorithms for CSP with 3-sequences [Gramm et al. 2001, Liu et al. 2011], in both papers presented algorithms in linear time. Meneses obtains optimal solutions via integer programming [Meneses et al. 2004, Meneses et al. 2005], In [Vilca 2013] solves this problem by cutting planes algorithm.

The CSP is defined as follows: Given a finite set $S = \{s^1, s^2, \ldots, s^n\}$ of strings with alphabet Σ, each string with length m, find a center string t of length m minimizing d, such that for every string $s^i \in S$, $d_H(t, s^i) \leq d$, we mean the Hamming distance between t and s^i. This is a NP-hard problem according to Frances and Litman[Frances and Litman 1997]. In the following, we show a formal proof of correctness from an efficient linear time algorithm for CSP instances with 3-sequences, computational experiments are reported, and finally, concluding remarks are presented.

2. Efficient linear time algorithm for 3-sequences

The 3-CSP-A algorithm is designed based on the isomorphic instance [Gramm et al. 2001]. An exact algorithm for 3-CSP with alphabet $\Sigma = 2$ is found in [Liu et al. 2011].

Definition 1 (Normalized instance) *Let S an instance, that is, $S = \{s^1, \ldots, s^n\}$, where $|s^i| = m$, with $1 \leq i \leq n$ and $1 \leq j \leq m$. Let $M_{n\times m}$ a matrix of characters from S each*

column is a position of the strings in S, so we have $M[c_1, c_2, \ldots, c_m]$

$$\phi(c'_j)_{1\leq i\leq n, 1\leq j\leq m} = \begin{cases} \lambda_1 & \textit{if } \sigma_1 \in c_j : max_{\sigma_i}|c^i_j = \sigma_1| \\ \lambda_2 & \textit{else if } \sigma_2 \in c_j : max_{\sigma_i}|c^i_j \setminus \sigma_1 = \sigma_2| \\ \ldots & \\ \lambda_k & \textit{else if } \sigma_k \in c_j : max_{\sigma_i}|c^i_j \setminus \{\sigma_1, \ldots, \sigma_{k-1}\} = \sigma_k| \end{cases}$$

Let $M'[c'_1, \ldots, c'_m]$*, and* S' *an instance based on characters from* M'*, as a result* S' *is called normalized instance from* S*.*

Theorem 1 *Let a CSP instance with 3-sequences, which denotes 3-CSP, that is,* $S = \{s^i \in \Sigma^m, i = 1, 2, 3\}$ *with alphabet* $|\Sigma| > 2$*, so the 3-CSP-A algorithm always finds an exact solution to 3-CSP.*

Proof 1 *Let* $S = \{s^1, s^2, s^3\}$ *a 3-CSP instance, let* ϕ *a bijective function that transforms* S *into* S'*, called normalized instance that is equivalent to* S *for* ϕ *according to [Gramm et al. 2001]. Let* $M_{3\times m}$ *a matrix of characters from* S *each column is a position of the strings in* S*, so we have* $M[c_1, c_2, \ldots, c_m]$ *with* m *length of strings in* S

$$\phi(c_j)_{1\leq i\leq 3, 1\leq j\leq m} = \begin{cases} a & \textit{if } \sigma_1 \in c_j : max_{\sigma_i}|c^i_j = \sigma_1| \\ b & \textit{else if } \sigma_2 \in c_j : max_{\sigma_i}|c^i_j \setminus \sigma_1 = \sigma_2| \\ c & \textit{else if } \sigma_3 \in c_j : max_{\sigma_i}|c^i_j \setminus \{\sigma_1, \sigma_2\} = \sigma_3| \end{cases}$$

We assume wlog five different cases in c_i*, that is,* $v_1 = aaa$*,* $v_2 = baa$*,* $v_3 = aba$*,* $v_4 = aab$ *and* $v_5 = abc$*. In order to get an optimal solution, divide* $M'[c_1, c_2, \ldots, c_m]$ *in blocks of 3-length and 2-length, consider all possible combinations of* $\{v_2, \ldots, v_5\}$*, then drop* v_1 *since it is a trivial case.*

In the case of 3-length blocks with repetition columns we have, $d_H(v_iv_jv_k, baa) = 2$ *and without repetitions* $d_H(v_iv_jv_k, aaa) = 1$ *for* $2 \leq i, j, k \leq 4$ *now inserting* v_5 *in our analysis with column repetition* $d_H(\{v_5v_iv_j, v_iv_5v_j, v_iv_jv_5\}, aaa) = 2$*, and finally* $d_H(v_5v_5v_5, abc) = 2$*.*

In the case of 2-length blocks we have, $d_H(v_iv_j, aa) = 1$ *and* $d_H(v_iv_i, ab) = 1$ *with* $2 \leq i, j \leq 4$ *after add* v_5*,* $d_H(v_2v_5, aa) = 1$*,* $d_H(v_3v_5, ab) = 1$*,* $d_H(v_4v_5, ac) = 1$*, and finally* $d_H(v_5v_5, aa) = 2$*.*

So in truth we are interested in the cases when the Hamming distance is equals to 1. Let l_j *a number that accounts for each time the case* v_j *is repeated where* $j \in \{1, 2, \ldots, 5\}$*, let* t' *a string of size* m*, an optimal solution of* S'*, in order to minimize the maximum Hamming distance* $d_H(t', S')$*, we verified various reductions for each case* v_j *presented. We are checking each kind of reductions for 2-length and 3-length blocks.*

Assume wlog $l_2 \leq l_3 \leq l_4$*, then* $l_3 = l_3 - l_2$*,* $l_4 = l_4 - (l_2 + l_3)$*, after these calculations, one of them is greater or equal to zero. let* $\rho_{iab} = \lfloor \frac{1}{2} l_i \rfloor$ *for* $2 \leq i \leq 4$

- *If* l_5 *mod* $3 > 0$ *and* $\{l_2, l_3, l_4\}$ *mod* $2 > 0$ *then* $\{\rho_{5ac}, \rho_{ab}, \rho_{aa}\} = 1$
- *If* l_5 *mod* $2 > 0$ *then* $\rho_{5ab} = 1$

Let t *a string that represents an optimal solution of 3-CSP-A, with* $1 \leq i \leq m$*, we*

have:

$$\begin{array}{llll}
s_i^1 = s_i^2 & & & t_i = s_i^1 = s_i^2 \\
s_i^1 = s_i^2 & s^2 \neq s^3 & \rho_{2ab} > 0 & t_i = s_i^3 \\
s_i^1 \neq s_i^2 & s_i^1 = s_i^3 & \rho_{3ab} > 0 & t_i = s_i^2 \\
s_i^1 \neq s_i^2 & s_i^1 \neq s_i^3 \quad s_i^2 = s_i^3 & \rho_{4ab} > 0 & t_i = s_i^1 \\
s_i^1 \neq s_i^2 & s_i^2 \neq s_i^3 \quad s_i^2 \neq s_i^3 & \rho_{5ab} > 0 & t_i = s_i^2 \\
s_i^1 \neq s_i^2 & s_i^2 \neq s_i^3 \quad s_i^2 \neq s_i^3 & \rho_{5ac} > 0 & t_i = s_i^3 \\
s_i^1 \neq s_i^2 & s_i^2 \neq s_i^3 \quad s_i^2 \neq s_i^3 & \rho_{ab} > 0 & t_i = s_i^2 \\
s_i^1 \neq s_i^2 & s_i^2 \neq s_i^3 \quad s_i^2 \neq s_i^3 & \rho_{aa} > 0 & t_i = s_i^1
\end{array}$$

Thus the theorem holds.

Example 1 *Let S_1 a 3-CSP instance, and let S_2 its normalized instance, we have:*

$$S_1 = \begin{cases} AGTATTGGTG \\ CCCTTTGAGA \\ TAGTGGGTCT \end{cases} \qquad S_2 = \begin{cases} aaabaaaaaa \\ bbbaaaabbb \\ cccabbaccc \end{cases}$$

After using the 3-CSP-A algorithm we have an optimal solution for the normalized instance S_2, that is, t = caccaaaaaa, with Hamming distance $d_H(t, S_2) = 4$.

Algorithm 1: Linear time algorithm 3-CSP-A, for 3-sequences.

```
Procedure Algorithm 3-CSP-A (n, m, S)
n : number of strings
m : size of strings
S : a 3-CSP instance, that is, S = {s^1, s^2, s^3}
Input  : Normalized Instance S = {s^1, s^2, s^3}
Output: optimal solution t ∈ Σ^m : d_H(t, S) ≤ d
// Let v_i: number of times that {aaa, aab, aba, baa, abc} appears in the jth column with 1 ≤ i ≤ 5 and 1 ≤ j ≤ m
if |S| = 3 then
    smallest ← getSmallest(v_2, v_3, v_4);
    // subtract the lowest value for cases: {aab,aba,baa}
    v_2 ← v_2 − smallest;
    v_3 ← v_3 − smallest;
    v_4 ← v_4 − smallest;
    for i=1 to m do
        if s_i^1 = s_i^2 then
            t_i ← s_i^1;
        end if
        else if s_i^2 = s_i^3 then
            t_i ← s_i^2;
            else if s_i^1 = s_i^3 then
                t_i ← s_i^1;
                // when all the characters are different, case: abc
                else if v_4 > 0 then
                    t_i ← s_i^1; v_4 ← v_4 − 1;
                    else if v_3 > 0 then
                        t_i ← s_i^2; v_3 ← v_3 − 1;
                    else if v_2 > 0 then
                        t_i ← s_i^3; v_2 ← v_2 − 1;
                    else
                        t_i ← s_i^j; //where j ∈ {1, 2, 3} one of them each time
                end if
            end if
        end if
    end for
end if
```

Instance	2 Characters			4 Characters			20 Characters		
m	seed	Val	time(s)	seed	Val	time(s)	seed	Val	time(s)
100	542319	27	< 1	791034	48	< 1	691425	63	< 1
200	7121	52	< 1	52151	91	< 1	351554	124	< 1
300	64874	77	< 1	68724	132	< 1	98121	183	< 1
400	6487	112	< 1	193185	180	< 1	246754	244	< 1
500	94115	124	< 1	15364	215	< 1	658745	311	< 1
600	5419	153	< 1	5419	260	< 1	154525	373	< 1
700	43212	180	< 1	524514	306	< 1	487754	437	< 1
800	2454	215	< 1	55364	354	< 1	754812	494	< 1
900	645387	234	< 1	6487	389	< 1	722451	557	< 1
1000	94315	260	< 1	153364	444	< 1	34567	616	< 1
2000	264554	508	< 1	6487	881	1	65743	1219	1
3000	68174	765	2	2454	1305	2	4432	1857	1
4000	4212	1003	3	53214	1760	3	543	2484	3
5000	722312	1278	5	934145	2188	6	344	3099	5

Table 1. Linear time algorithm 3-CSP-A, for 3-sequences.

3. Computational Experiments and Results

Some computational experiments was done involving forty two 3-CSP instances, using the instance generator described in the literature [Meneses et al. 2004]. These instances consider alphabets with two, four and twenty characters, which obtained optimal solutions using the 3-CSP-A algorithm presented in the Section 2. The header on Table 2 has the following meanings: the first column indicates the tested instance, indicating the parameters (m) size of string; the columns (2,4, and 20 Characters) indicates the (seed) number to generate a random instance, (Val) values of optimal solutions to 3-sequences for binary, DNA, and protein types as well as their execution times in seconds.

4. Final Remarks

We proposed an exact algorithm for the special case of CSP with 3-sequences and alphabet size $|\Sigma| > 2$,given the corresponding theoretical analysis.

References

Frances, M. and Litman, A. (1997). On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119.

Gramm, J., Niedermeier, R., and Rossmanith, P. (2001). Exact solutions for closest string and related problems. In *Proceedings of the 12th International Symposium on Algorithms and Computation*, ISAAC '01, pages 441–453.

Liu, X., Liu, S., Hao, Z., and Mauch, H. (2011). Exact algorithm and heuristic for the closest string problem. *Computers & Operations Research*, 38(11):1513–1520.

Meneses, C., Lu, Z., Oliveira, C., and Pardalos, P. (2004). Optimal solutions for the closest string problem via integer programming. *INFORMS Journal on Computing*, 16(4):419–429.

Meneses, C., Oliveira, C., and Pardalos, P. (2005). Optimization techniques for string selection and comparison problems in genomics. *Engineering in Medicine and Biology Magazine, IEEE*, 24(3):81–87.

Vilca, O. L. (2013). Métodos para problemas de seleção de cadeias de caracteres. Master's thesis, Universidade Federal do ABC, Santo André, São Paulo.