J Comb Optim
DOI 10.1007/s10878-015-9914-6

Constrained pairwise and center-star sequences alignment problems

Yong Zhang[1,3] · Joseph Wun-Tat Chan[2] · Francis Y. L. Chin[3] · Hing-Fung Ting[3] · Deshi Ye[4] · Feng Zhang[5] · Jianyu Shi[6]

Abstract Sequence alignment is a fundamental problem in computational biology, which is also important in theoretical computer science. In this paper, we consider the problem of aligning a set of sequences subject to a given constrained sequence.

A preliminary version of this paper appeared in the Proceedings of the 8th International Frontiers of Algorithmics Workshop (FAW 2014) Lecture Notes in Computer Science, Volume 8497, 2014, pp 309–319.

✉ Deshi Ye
yedeshi@zju.edu.cn

Yong Zhang
zhangyong@siat.ac.cn

Joseph Wun-Tat Chan
cswtchan@gmail.com

Francis Y. L. Chin
chin@cs.hku.hk

Hing-Fung Ting
hfting@cs.hku.hk

Feng Zhang
amyfzhang@gmail.com

Jianyu Shi
jianyushi@nwpu.edu.cn

1 Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China

2 College of International Education, Hong Kong Baptist University, Kowloon, Hong Kong, China

3 Department of Computer Science, The University of Hong Kong, Pok Fu Lam, Hong Kong, China

4 College of Computer Science, Zhejiang University, Hangzhou, China

5 College of Mathematics and Information Science, Hebei University, Baoding, China

6 School of Life Science, Northwestern Polytechnical University, Xi'an, China

Published online: 06 June 2015

Given two sequences $A = a_1a_2 \dots a_n$ and $B = b_1b_2 \dots b_n$ with a given distance function and a constrained sequence $C = c_1c_2 \dots c_k$, our goal is to find the optimal sequence alignment of A and B w.r.t. the constraint C. We investigate several variants of this problem. If $C = c^k$, i.e., all characters in C are same, the optimal constrained pairwise sequence alignment can be solved in $O(\min\{kn^2, (t - k)n^2\})$ time, where t is the minimum number of occurrences of character c in A and B. If in the final alignment, the alignment score between any two consecutive constrained characters is upper bounded by some value, which is called GB-CPSA, we give a dynamic programming with the time complexity $O(kn^4/\log n)$. For the constrained center-star sequence alignment (CCSA), we prove that it is NP-hard to achieve the optimal alignment even over the binary alphabet. Furthermore, we show a negative result for CCSA, i.e., there is no polynomial-time algorithm to approximate the CCSA within any constant ratio.

Keywords Sequence alignment · Dynamic programming · Complexity

1 Introduction

Sequence alignment (Mount 2004) is a fundamental method in computational biology to analyzing the functional, structural, or evolutionary relationships between a set of DNA, RNA, or protein sequences. It is well-defined and had received in-depth study in bioinformatics and theoretical computer science during these years.

Let Σ be the set of alphabets. For s, a sequence of n characters over Σ, $s[x \dots y]$ denotes the substring of s from the xth character to the yth character of s, where $1 \leq x < y \leq n$. In particular, let $s[x]$ denote the xth character of s. In general, the size of the alphabet set is not large. In this paper, we assume that $|\Sigma| = c$, where c is a constant. E.g., in DNA sequence, $\Sigma = \{A, C, G, T\}$.

Given two sequences s and t, the *Pairwise Sequence Alignment* (PSA) w.r.t. s and t is two sequences s' and t' such that s' and t' have the same length n' and removing all space characters "$-$" from s' and t' gives s and t respectively. Let function α define the alignment positions, i.e., $\alpha(s, i)$ denotes the position of the character in s' corresponds to character $s[i]$. So $s'[\alpha(s, i)] = s[i]$. For a given distance function $\delta(x, y) \geq 0$ which measures the mutation distance between two characters, where $x, y \in \Sigma \cup \{-\}$, the pair-wise score of two length n' sequences s' and t' is defined as $\sum_{1 \leq x \leq n'} \delta(s'[x], t'[x])$. Usually, $\delta(x, x) = 0$ for any $x \in \Sigma$. The optimal pairwise sequence alignment with the minimum alignment score can be solved in $O(n^2)$ time by dynamic programming, where n is the length of the longer sequence. The problem of finding the longest common subsequence (LCS) of two sequences is a special case of the sequence alignment, which is well-studied for many years. Note that the target of the LCS problem is to find the maximized length of a common subsequence of two strings, however, this maximization problem can be converted to the minimization problem by defining the distance function $\delta(\cdot, \cdot)$, such that $\delta(x, -) = 1$, $\delta(-, y) = 1$, $\delta(x, y) = 2$ if $x \neq y$ and $\delta(x, y) = 0$ if $x = y$, where both x and y are characters in these two strings and not $'-'$. The LCS problem can be solved in $O(n^2)$ time (Cormen et al. 2009). The only subquadratic algorithm for LCS was given by Masek and Paterson (1980), which runs in $O(n^2/\log n)$ if the size of the alphabet set is bounded by a constant. Iliopoulos and Sohel Rahman (2008) introduced the LCS problem with fixed gap (FIG), in which the distance between any two consecutive characters in the common subsequence in both input sequences are bounded by a given parameter K. They presented an $O(n^2 + R\log\log n)$ algorithm, where R is the total number of matches between the input sequences.

In the *multiple sequence alignment* (MSA) problem, we are given m sequences $S = \{s_1, s_2, \ldots, s_m\}$ with maximum length n. The output of an MSA is an alignment matrix A, with m rows and $n'(\geq n)$ columns of characters over $\Sigma \cup \{-\}$, such that removing space characters from the ith row of A gives s_i for $1 \leq i \leq m$. The sum-of-pairs (SP) score of the MSA A is defined to be the sum of the pair-wise scores of all pairs of the sequences, i.e.,

$$score(A) = \sum_{1\leq i<j\leq m} \sum_{1\leq p\leq n'} \delta(A_{i,p}, A_{j,p})$$

where $A_{i,p}$ and $A_{j,p}$ are the characters at the i- and jth row and the pth column of A, respectively. Star alignment is also an important alignment in computational biology (Setubal and Meidanis 1997). In the star alignment, a center sequence s shall be identified, and it is used as the "center of star" when aligning with all other sequences. A sequence is a center if it is the most similar to all the rest using pairwise alignment. It was shown in Bonizzoni and Vedova (2001) and Wang and Jiang (1994) that finding an alignment matrix with the minimum sum-of-pair alignment score for $m \geq 3$ sequences is NP-Hard. There are a number of heuristics which approximate the optimal alignment, some with guaranteed worst case approximation ratio, and some with good performance in practice, e.g., BLAST (1990), Clustal W (2007). One of the approximation algorithms in Gusfield (1993), based on the center-star alignment, can approximate the optimal alignment within a factor of $2-2/m$ in $O(mn^2)$ time, where m is the number of sequences.

According to the biology knowledge, some residues in the multiple sequence must be aligned in the same columns. For example, His12, Lys41 and His119 should be aligned in the same columns when consider the alignment of RNase sequence. That motivates to study the *constrained sequence alignment problem*, which was introduced by Tang et al. (2003). In the *constrained multiple sequence alignment problem* (CMSA), we are given, in addition to the inputs of the multiple sequence alignment problem, a constrained sequence $C = c_1c_2\ldots c_k$, where C is a common subsequence of all $s_i \in \{s_1, s_2, \ldots, s_m\}$. The solution of a CMSA problem is a constrained alignment matrix A which is an alignment matrix such that each character in C appears in an entire column of A and also in the same order, i.e. there exists a list of integers $\{g_1, g_2, \ldots, g_k\}$ where $1 \leq g_1 < \cdots < g_k \leq n'$ and for all $1 \leq i \leq m$ and for all $1 \leq j \leq k$, we have $A_{i,g_j} = c_j$.

Let A be a CMSA matrix for $S = \{s_1, s_2, \ldots, s_m\}$ and the constrained sequence C. Define $score(A)$ to be the score of the CMSA matrix A. Let $A^*_{S,C}$ be the optimal CMSA matrix and $A_{S,C}$ be the CMSA matrix derived by an approximation algorithm. The approximation algorithm is said to have an approximation ratio r if and only if for any S and C,

$$\frac{score(A_{S,C})}{score(A^*_{S,C})} \leq r.$$

For two sequences of a CMSA, it is called *constraint pairwise sequence alignment* (CPSA), Tang et al. (2003) gave an algorithm with both the time and space complexity $O(kn^4)$, where k is the length of the constrained sequence. The result was improved by Chin et al. (2004) to $O(kn^2)$. To approximate the optimal constrained alignment for $m \geq 3$ sequences, Chin et al. (2005) introduced the *constrained center-star sequence alignment* (CCSA). Similar to the center-star sequence alignment to approximate the multiple sequences alignment, the optimal constrained center-star sequence alignment has an approximation ratio of no more than $2-2/m$. However, the proposed algorithm for finding the optimal constrained center-star sequence alignment (CCSA) for m sequences takes $O(Cmn^2)$ time, where C is the total number of occurrences of the constrained sequence in the m sequences. As C can be exponential, the time complexity for CCSA can be exponential.

In this paper, we consider various constrained sequence alignment problems. Note that *the constrained pairwise sequence alignment* can be solved in $O(kn^2)$ Chin et al. (2004). In this work, we give an $O((t-k)n^2)$ algorithm to solve the CPSA problem in which the constrained sequence $C = c^k$, and t is the minimum number of occurrences of character c in the two sequences. Combined with the previous result, this variant can be solved in $O(\min\{kn^2, (t-k)n^2\})$ time when $k < t$ and $O(n^2)$ when $k = t$. Next, we study another variant of the constrained pairwise sequence alignment, say *gap-bounded constrained pairwise sequence alignment* (GB-CPSA), in which the alignment score between any two consecutive constrained characters is upper bounded by some value in the final alignment. For the GB-CPSA, we give a dynamic programming with the time complexity $O(kn^2 + \sum_{p=1}^{k} R_{p-1}R_pn^2/\log n) = O(kn^4/\log n)$, where R_p is the number of matches on c_p in A and B. Finally, we show that *the constrained center-star sequence alignment* problem for multiple sequences is NP-hard even if the size of the alphabet set is two, and this problem cannot be approximated within a constant factor for some distance function.

In the remainder of this paper, we first consider two variants of the constrained pairwise sequence alignment in Sect. 2. The we turn to study the complexity of the constrained center-star sequence alignment problem in Sect. 3. The conclusion is given in Sect. 4.

2 Constrained pairwise sequence alignment

In this section we study the constrained pairwise sequence alignment (CPSA) problem. Given two sequences $A = a_1a_2 \dots a_n$ and $B = b_1b_2 \dots b_n$, a constrained sequence $C = c_1c_2 \dots c_k$, and a distance function δ, the CPSA problem is to find the minimum-score PSA, A' and B', such that A' and B' have the same length $n' \geq n$ and $A'[g_i] = B'[g_i] = c_i$ for $1 \leq i \leq k$ and some integers $1 \leq g_1 < g_2 < \cdots < g_k \leq n'$. The distance function $\delta(x, y)$ is a score of aligning the character x and the character y, where $x, y \in \Sigma \cup \{-\}$. The distance function δ always satisfies $\delta(x, x) = 0$ for any $x \in \Sigma \cup \{-\}$.

Let $S(i, j, p)$ denote the optimal CPSA score for the sequences $A[1 \ldots i]$ and $B[1 \ldots j]$ with the constrained sequence $C[1 \ldots p]$ under the distance function δ. The function $S(i, j, p)$ can be described recursively as follows as in Chin et al. (2004).

$$S(i,j,p) = \min \begin{cases} S(i-1, j-1, p-1) & \text{if } a_i = b_j = c_p \\ S(i-1, j-1, p) + \delta(a_i, b_j) & \text{if } i, j > 0 \\ S(i-1, j, p) + \delta(a_i, -) & \text{if } i > 0 \\ S(i, j-1, p) + \delta(-, b_j) & \text{if } j > 0 \end{cases}$$

For the boundary cases, $S(i, 0, 0) = S(0, j, 0) = 0$ and $S(0, j, \ell) = S(i, 0, \ell) = \infty$ for $0 \le i, j \le n$ and $0 < \ell \le k$. The last equation means there is no way to align an empty sequence with a non-empty constraint sequence.

This algorithm computes all the entries $S(i, j, p)$ in $O(kn^2)$ time because $i, j \le n$ and $p \le k$ and each entry can be computed in constant time.

2.1 Constrained sequence $C = c^k$

In this part, we consider a special case where $c_1 = c_2 = \ldots = c_k$, i.e., the k characters are all the same. Let that character be c and denote $C = c^k$. Without loss of generality, assume that the number of occurrences of character c in B is t and it is no more than the number of occurrences of c in A. Consider the case when k approaches t, previous algorithm (Chin et al. 2004) takes $O(tn^2)$ time. In the following we give a more efficient algorithm when k is close to t. Our algorithm takes $O(n^2)$ time when $k = t$ and $O((t - k)n^2)$ time when $k < t$.

Now we show how to find the optimal PSA score in $O((t-k)n^2)$ time. Let $T(i, j, q)$ denote the optimal PSA score for sequences $A[1 \ldots i]$ and $B[1 \ldots j]$ and the distance function δ such that in the PSA, A' and B', there are exactly q occurrences of character c in B' matched with a character (in A') other than c. Precisely, there are exactly q integers $g_1, g_2, \ldots, g_q$ with $B'[g_i] = c$ and $A'[g_i] \ne c$. The function $T(i, j, q)$ can be defined recursively as follows:

$$T(i,j,q) = \min \begin{cases} T(i-1, j-1, q) + \delta(a_i, b_j) & \text{if } b_j \ne c \text{ or } a_i = b_j = c \\ T(i-1, j-1, q-1) + \delta(a_i, b_j) & \text{if } b_j = c \text{ and } a_i \ne c \\ T(i-1, j, q) + \delta(a_i, -) & \text{if } i > 0 \\ T(i, j-1, q-1) + \delta(-, c) & \text{if } b_j = c \\ T(i, j-1, q) + \delta(-, b_j) & \text{if } b_j \ne c \end{cases}$$

In the recursive step, there are three choices for the optimal PSA to proceed. (Note that the first and the second cases, the fourth and the fifth cases are mutually exclusive, respectively.) The optimal PSA could either (i) match a_i with b_j (ii) match a_i with space, or (iii) match b_j with space. Obviously, the optimal PSA should yield the minimum score among these three choices. For (i), the PSA would include the distance between a_i and b_j and proceed with shortened inputs $A[1 \ldots i-1]$ and $B[1 \ldots j-1]$. In case of $b_j \ne c$, or $a_i = b_j = c$, there are exactly q occurrences (otherwise $q - 1$) of character c in $B[1 \ldots j-1]$ will match with a character other than c. For (ii), the

PSA would include the distance between a_i and the space character "–". Since only a_i is matched but not b_j, the PSA proceeds with $A[1 \dots i-1]$ and $B[1 \dots j]$ and it still requires exactly q occurrences of character c in $B[1 \dots j]$ matched with a character other than c. For (iii), the PSA would include the distance between b_j and the space character "–" and proceed with $A[1 \dots i]$ and $B[1 \dots j-1]$. However, it differs for the case where $b_j = c$ and $b_j \neq c$. If $b_j = c$, we have this character c matched with a character "–", which is not c. Thus the subsequent PSA requires only $q-1$ occurrences of character c in $B[1 \dots j-1]$ matched with a character other than c. Otherwise, it still requires exactly q occurrences of character c in $B[1 \dots j-1]$ matched with a character other than c.

Remind that in this section our target is to find the CPSA with constrained sequence c^k. In fact, it is equivalent to find the minimum-score PSA among the PSA with exactly i occurrences of character c in B matched with a character other than c for $i = 0$ to $t-k$. Therefore, the required CPSA is the minimum-score PSA among those correspond to $T(m, n, i)$ for $0 \leq i \leq t-k$. In our algorithm we can compute the entries $T(i, j, q)$ for all $i, j \leq n,\ q \leq t-k$. Since each entry can be computed in constant time, the algorithm can find CPSA in $O((t-k)n^2)$ time.

Combining the algorithm of Chin et al. (2004) with time complexity $O(kn^2)$ and our new algorithm with time complexity $O((t-k)n^2)$ if $k < t$ and $O(n^2)$ if $k = t$, we can compute the required CPSA efficiently as shown in the following theorem.

Theorem 1 *The CPSA problem for two sequences with length n, the constrained sequence of c^k, and a given distance function can be solved in $O(\min\{k, t-k\}n^2)$ if $k < t$ and in $O(n^2)$ if $k = t$ where t is the minimum number of occurrences of character c in these two sequences.*

2.2 Gap-bounded constrained pairwise sequence alignment

In this part, we consider the gap-bounded constrained pairwise sequence alignment (GB-CPSA), i.e., given two sequences $A = a_1 a_2 \dots a_n$, $B = b_1 b_2 \dots b_n$, the constrained sequence $C = c_1 c_2 \dots c_k$ and a distance function δ, the GB-CPSA is to find a minimum-score PSA, A' and B', such that A' and B' have the same length $n' \geq n$, $A'[g_i] = B'[g_i] = c_i$ for some integers $1 \leq g_1 < g_2 < \cdots < g_k \leq n'$ and the alignment score between $A'[g_i + 1 \dots g_{i+1} - 1]$ and $B'[g_i + 1 \dots g_{i+1} - 1]$ is upper bounded by K. Roughly speaking, any two consecutive characters in the constrained sequence are not far away in the final alignment. Different from FIG (Iliopoulos and Sohel Rahman 2008), our problem focuses on the bounded gap of the consecutive characters in the constrained sequence, but not in the common subsequence. Between two consecutive constrained characters in the final alignment, some common characters may be aligned together. Due to the distribution of the constrained characters in the input sequences A and B, there may not exist a PSA to satisfy the gap-bounded property. In this case, the algorithm should output "No alignment".

Let $S(s, t)$ be the minimum alignment score w.r.t. sequences s and t. Let $S_1(i, j, p)$ be the minimum score of GB-CPSA w.r.t. $A[1 \dots i]$, $B[1 \dots j]$ and the constrained sequence $C[1 \dots p]$ such that the constrained character c_p is aligned with a_i and b_j. $S_2(i, j, p)$ is defined similarly but the constrained character c_p is not aligned with a_i

and b_j. Note that $S_1(i, j, 0)$ and $S_2(i, j, 0)$ are the traditional alignment score without considering the constrained sequence. In details, $S_1(i, j, p)$ and $S_2(i, j, p)$ are given as follows.

- $S_1(i, j, p)$
  - If $a_i \neq c_p$ or $b_j \neq c_p$, $S_1(i, j, p) = +\infty$.
  - If $a_i = b_j = c_p$, the value of $S_1(i, j, p)$ depends on the previous values $S_1(i', j', p-1)$ and the alignment between $A[i'+1 \dots i-1]$ and $B[j'+1 \dots j-1]$ such that the alignment score between c_{p-1} and c_p is bounded by K. Formally,

$$S_1(i, j, p) = \min_{i', j'} \{S_1(i', j', p-1) + S(A[i'+1 \dots i-1], B[j'+1 \dots j-1])\} + \delta(c_p, c_p)$$

  where $a_{i'} = b_{j'} = c_{p-1}$ and $S(A[i'+1 \dots i-1], B[j'+1 \dots j-1]) \leq K$.
- $S_2(i, j, p)$
  Since the constrained character c_p is aligned before a_i and b_j, the score function follows the traditional computation for sequence alignment, i.e.,

$$S_2(i, j, p) = \min \begin{cases} \min\{S_1(i-1, j-1, p), S_2(i-1, j-1, p)\} + \delta(a_i, b_j) & \text{if } i, j > 0 \\ \min\{S_1(i-1, j, p), S_2(i-1, j, p)\} + \delta(a_i, -) & \text{if } i > 0 \\ \min\{S_1(i, j-1, p), S_2(i, j-1, p)\} + \delta(-, b_j) & \text{if } j > 0 \end{cases}$$

After all entries of $S_1(i, j, p)$ and $S_2(i, j, p)$ are computed for $1 \leq i, j \leq n$ and $1 \leq p \leq k$, the minimum score of the GB-CPSA is

$$\min\{S_1(n, n, k), S_2(n, n, k)\}.$$

There are kn^2 entries in $S_1(i, j, p)$ and $S_2(i, j, p)$, respectively. The recursive formula of $S_1(i, j, p)$, $S_2(i, j, p)$ can be determined in constant time given previous values of $S_1()$ and $S_2()$. To compute the value of $S_1(i, j, p)$, the previous alignments on c_{p-1} with the bounded gap to c_p have to be considered. Thus, we must do the following two jobs:

(1) efficiently find the alignments such that of $a_{i'} = b_{j'} = c_{p-1}$, and
(2) efficiently compute the alignment score between $A[i'+1 \dots i-1]$ and $B[j'+1 \dots j-1]$.

For the first job, we use two arrays to store the sorted positions of each character c_p in A and B, respectively. By scanning the sequences A and B once, the arrays for c_p $(1 \leq p \leq k)$ can be constructed in $O(n)$. When computing $S_1(i, j, p)$, we should determine the positions i' and j' such that $a_{i'} = b_{j'} = c_{p-1}$. Given i and j, the largest i' and j' with $i' < i$, $j' < j$ and $a_{i'} = b_{j'} = c_{p-1}$ can be achieved in $O(\log n)$ time by binary search. Whereas $(a_{i'}, b_{j'})$ is the possible closest match satisfying the constrained character c_{p-1} w.r.t. $S_1(i, j, p)$. Each of the previous matches can be found in $O(1)$ time by tracing the arrays for c_{p-1} in A and B.

For the second job, the $O(n^2/\log n)$ time algorithm from Masek and Paterson (1980) can be applied to our problem because we do not need to consider the constrained characters and this job is computing the alignment score of two strings with a constant size of alphabets set. The value of $S(A[i'+1 \dots i-1], B[j'+1 \dots j-1], \phi)$ can be achieved in $O((i-i')(j-j')/\log n)$ time, which is upper bounded by $O(n^2/\log n)$.

Let R_p be the total number of matches on c_p in A and B. Thus, given previous values, $S_1(i, j, p)$ can be computed in

$$O(\log n + R_{p-1}n^2/\log n) = O(R_{p-1}n^2/\log n).$$

For c_p, there are n^2 entries of $S_1(i, j, p)$, besides these entries such that $a_i = b_j = c_p$, the values of all other entries can be determined to be $+\infty$ in constant time. Therefore, the complexity to fill all $S_1(i, j, p)$ entries w.r.t. c_p is

$$O(R_{p-1}R_p n^2/\log n).$$

Considering all constrained characters in $C = c_1c_2\dots c_k$, filling all entries of $S_1(i, j, p)$ takes time

$$O\left(\sum_{p=1}^{k} R_{p-1}R_p n^2/\log n\right).$$

From above analysis, we conclude the following theorem.

Theorem 2 *The Gap-Bounded Constrained Pairwise Sequence Alignment (GB-CPSA) can be solved in* $O(kn^4/\log n)$.

Proof The time complexity to compute all entries of $S_1(i, j, p)$ is $O\left(\sum_{p=1}^{k} R_{p-1}R_p n^2/\log n\right)$. The time complexity to compute all entries of $S_2(i, j, p)$ is $O(kn^2)$. Determine the minimum score takes $O(1)$ time by comparing $S_1(n, n, k)$ and $S_2(n, n, k)$. Thus, the time complexity for solving GB-CPSA is

$$O\left(kn^2 + \sum_{p=1}^{k} R_{p-1}R_p n^2/\log n) = O(kn^4/\log n\right).$$

□

Remark When compute each entry of $S_1(i, j, p)$ and $S_2(i, j, p)$, we may use two tables $T_1(i, j, k)$ and $T_2(i, j, k)$ to store the previous entry leads to the minimum value. Thus, for the GB-CPSA problem, the sequence alignment with the minimum score can be achieved in linear time by tracing back from $T_1(i, j, p)$ and $T_2(i, j, p)$.

3 Constrained center-star sequence alignment

In this section, we study the problem of *constrained center-star sequence alignment (CCSA)*. This problem is similar to the CMSA problem since the goal is also to find an optimal alignment matrix with a constrained sequence. However, CCSA has a different way of calculating the score of an alignment matrix. In CMSA, we minimize the sum of all pairwise scores of the alignment matrix. In CCSA, we minimize, for $s \in S$, the sum of the pairwise scores of s with every sequence in $S - \{s\}$. Precisely speaking, the score of an alignment matrix A in CCSA is defined to be

$$\min_{1 \le i \le m} \sum_{1 \le j \le m, j \ne i} \sum_{1 \le p \le n'} \delta(A_{i,p}, A_{j,p}).$$

For an optimal alignment matrix A, we call a particular sequence s_ℓ the *center sequence* if the score of A is

$$\sum_{1 \le j \le m, j \ne \ell} \sum_{1 \le p \le n'} \delta(A_{\ell,p}, A_{j,p}).$$

3.1 NP-hardness of CCSA

In this part, we prove that CCSA is NP-Hard over binary alphabet, i.e., the size of alphabet set is 2. To prove this, we give a polynomial time reduction from the NP-hard problem *Maximum Independent Set (MIS)* (Garey and Johnson 1979) to CCSA. Consider the decision version of MIS as follows.

> Maximum Independent Set (MIS): Given a graph $G = (V, E)$ and an integer k, is there a subset $V' \subseteq V$ of vertices for $|V'| = k$ such that each edge in E is incident on at most one vertex in V'?

For any instance of the decision version of MIS which includes a graph $G = (V, E)$ and an integer k, we give a polynomial-time transformation to an instance of CCSA which includes the alphabet set Σ, a set of sequences S over Σ, a constrained sequence C, and a distance function δ. We define the transformation as Π, i.e., $\Pi(G, k) = (\Sigma, S, C)$. Let $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. The transformation $\Pi(G, k)$, constructed according to G and k, is shown as follows. We have $\Sigma = \{a, b\}$ and $S = \{t_1, \ldots, t_m, s_1, \ldots, s_m\}$, where $t_1 = \cdots = t_m = (aba)^n$ and each s_i is an encoding corresponding to an edge e_i. Suppose $e_i = (v_p, v_q)$ with $p < q$, then $s_i = (bab)^{p-1}(baa)(bab)^{q-p-1}(aab)(bab)^{n-q}$. The constrained sequence $c = b^k$. The distance function δ between any two characters in $\Sigma \cup \{-\}$ of the constructed CCSA instance is defined as follows.

	a	b	–
a	0	1	2
b	1	0	2
–	2	2	0

Consider $s_i = (bab)^{p-1}(baa)(bab)^{q-p-1}(aab)(bab)^{n-q}$ and $t_j = (aba)^n$ for $1 \leq i, j \leq m$ and $e_i = (v_p, v_q)$ where $p < q$. Two possible alignments of s_i and t_j are

$$\begin{aligned} s_i' &= \text{-}(bab)^{p-1}(baa)(bab)^{q-p-1}(aab)(bab)^{n-q} \\ t_j' &= (aba)^n\text{-} \end{aligned} \tag{1}$$

and

$$\begin{aligned} s_i' &= (bab)^{p-1}(baa)(bab)^{q-p-1}(aab)(bab)^{n-q}\text{-} \\ t_j' &= \text{-}(aba)^n. \end{aligned} \tag{2}$$

It is easy to check that both alignments yield a score of $n+3$. The following lemma shows that except the above two alignments all other alignments of s_i and t_i yield a score greater than $n+3$.

Lemma 1 *For any $1 \leq i, j \leq m$, the alignments of s_i and t_j according to Alignments (1) and (2) yield a score of $n+3$. All other alignments of s_i and t_j yield a score greater than $n+3$.*

Proof Suppose edge $e_i = (v_p, v_q)$, then $s_i = (bab)^{p-1}(baa)(bab)^{q-p-1}(aab)(bab)^{n-q}$. We also have $t_j = (aba)^n$.

We consider all possible alignments of s_i and t_j and divide them into 7 cases. These cases are characterized by how the sub-string "*baa*" in s_i "overlaps" with the pth "*aba*" in t_j. The positions of the characters of sub-string "*baa*" in s_i (as well as the pth "*aba*" in t_j) are $3p-2$, $3p-1$ and $3p$, respectively. The following cases consider all possible relationships between the values of $\alpha(s, 3p-2)$, $\alpha(s, 3p-1)$, $\alpha(s, 3p)$, $\alpha(t, 3p-2)$, $\alpha(t, 3p-1)$, and $\alpha(t, 3p)$.

Case (1) $\alpha(t, 3p-2) > \alpha(s, 3p)$.

In this case the corresponding sub-string "*baa*" in s_i appears before the pth "*aba*" in t_j in the alignment. We further consider the sub-case $\alpha(s, 3p) < \alpha(t, 3p-2) \leq \alpha(s, 3p+1)$. The other sub-case $\alpha(s, 3p+1) < \alpha(t, 3p-2)$ can be proved similarly. The alignment can be seen as breaking each of s_i and t_j into two parts and aligning corresponding parts of s_i and t_j as follows.

s_i	$(bab)^{p-1}\mathbf{baa}$	$(bab)^{q-p-1}(aab)(bab)^{n-q}$
t_j	$(aba)^{p-1}$	$\mathbf{aba}(aba)^{n-p}$

The first part of s_i, denote by s_i^1 has $p+1$ "*a*" and $2p-1$ "*b*", the first part of t_j, denote by t_j^1 has $2p-2$ "*a*" and $p-1$ "*b*". Even if all "*a*" in s_i^1 and all "*b*" in t_j^1 are matched, there are still $p-3$ "*a*" and p "*b*" unmatched. Therefore, the score of the first part of this alignment is at least $p+3$ because if all unmatched $p-3$ "*a*" align with $p-3$ "*b*" there are still 3 "*b*" that must align with space characters. For the same reason, the score of second part of this alignment is at least $n-p+4$. Hence, the total score of this pairwise alignment is at least $n+7$.

Case (2) $\alpha(s, 3p-1) < \alpha(t, 3p-2) \leq \alpha(s, 3p)$. The alignment can be seen as breaking each of s_i and t_j into two parts and aligning corresponding parts of s_i and t_j as follows.

s_i	$(bab)^{p-1}\mathbf{ba}$	$\mathbf{a}(bab)^{q-p-1}(aab)(bab)^{n-q}$
t_j	$(aba)^{p-1}$	$\mathbf{aba}(aba)^{n-p}$

Similarly to the counting in Case (1), we have the score of the first part of the alignment at least $p+2$ and the score of the second part of the alignment at least $n-p+2$. Hence, the total score of the alignment is at least $n+4$.

Case (3) $\alpha(s, 3p-2) < \alpha(t, 3p-2) \leq \alpha(s, 3p-1)$. The alignment can be seen as breaking each of s_i and t_j into two parts and aligning corresponding parts of s_i and t_j as follows.

s_i	$(bab)^{p-1}\mathbf{b}$	$\mathbf{aa}(bab)^{q-p-1}(aab)(bab)^{n-q}$
t_j	$(aba)^{p-1}$	$\mathbf{aba}(aba)^{n-p}$

In the first part, the minimal alignment score is $p+1$, and this happens only when $s_i^{1\prime} = (bab)^{p-1}b$ and $t_j^{1\prime} = \text{-}(aba)^{p-1}$. Otherwise, the score is greater than $p+1$. For the second part, consider the sub-string of "aa", the $(n-p-1)$ sub-strings of "bab", and the sub-string of "aab", which form s_i^2. The score of each of "aa" and the $(n-p-1)$ "bab" is at least 1. Moreover, there must be at least one space character to be inserted to s_i^2 for the alignment with t_j^2. So the minimum score is $n-p+2$ and it happens only when $s_i^{2\prime} = aa(bab)^{q-p-1}(aab)(bab)^{n-q}\text{-}$ and $t_j^{2\prime} = (aba)^{n-p+1}$. Thus the minimum score for aligning s_i and t_j in this case is $n+3$ and it happens only when $s_i^{\prime} = (bab)^{p-1}(baa)(bab)^{q-p-1}(aab)(bab)^{n-q}\text{-}$ and $t_j^{\prime} = \text{-}(aba)^{n}$. Otherwise, the score is greater than $n+3$.

Case (4) $\alpha(t, 3p-2) \leq \alpha(s, 3p-2)$ and $\alpha(t, 3p) \geq \alpha(s, 3p)$. We further consider the sub-case $\alpha(s, 3p-3) < \alpha(t, 3p-2) \leq \alpha(s, 3p-2)$ and $\alpha(s, 3p) \leq \alpha(t, 3p) < \alpha(s, 3p+1)$. The other sub-cases can be proved similarly. The alignment can be seen as breaking each of s_i and t_j into three parts and aligning corresponding parts of s_i and t_j as follows.

s_i	$(bab)^{p-1}$	**baa**	$(bab)^{q-p-1}(aab)(bab)^{n-q}$
t_j	$(aba)^{p-1}$	**aba**	$(aba)^{n-p}$

The alignment score for the first part is at least $p+3$, for the second part is at least 2, and for the third part is at least $n-p+2$. Thus the alignment score of s_i and t_j is at least $n+7$.

Case (5) $\alpha(t, 3p-2) \leq \alpha(s, 3p-2)$ and $\alpha(s, 3p-1) \leq \alpha(t, 3p) \leq \alpha(s, 3p)$. The alignment can be seen as breaking each of s_i and t_j into two parts and aligning corresponding parts of s_i and t_j as follows.

s_i	$(bab)^{p-1}\mathbf{ba}$	$\mathbf{a}(bab)^{q-p-1}(aab)(bab)^{n-q}$
t_j	$(aba)^{p-1}\mathbf{aba}$	$(aba)^{n-p}$

Similarly to that of Case (3), in the first part, the minimal alignment score is $p+1$, and this happens only when $s_i^{1\prime} = \text{-}(bab)^{p-1}ba$ and $t_j^{1\prime} = (aba)^{p-1}$. Otherwise, the score is greater than $p+1$. For the second part, the minimum alignment score is $n-p+2$ and it happens only when $s_i^{2\prime} = a(bab)^{q-p-1}(aab)(bab)^{n-q}$ and $t_j^{2\prime} = (aba)^{n-p}\text{--}$. Thus the minimum score for aligning s_i and t_j in this case is $n+3$ and it happens only when $s_i' = \text{-}(bab)^{p-1}(baa)(bab)^{q-p-1}(aab)(bab)^{n-q}$ and $t_j' = (aba)^n\text{-}$. Otherwise, the score is greater than $n+3$.

Case (6) $\alpha(t, 3p-2) \leq \alpha(s, 3p-2)$ and $\alpha(s, 3p-2) \leq \alpha(t, 3p) \leq \alpha(s, 3p-1)$. The alignment can be seen as breaking each of s_i and t_j into two parts and aligning corresponding parts of s_i and t_j as follows.

s_i	$(bab)^{p-1}\mathbf{b}$	$\mathbf{aa}(bab)^{q-p-1}(aab)(bab)^{n-q}$
t_j	$(aba)^{p-1}\mathbf{aba}$	$(aba)^{n-p}$

Similar to that in Case (2), the total score of this alignment is at least $n+4$.

Case (7) $\alpha(t, 3p) < \alpha(s, 3p-2)$. The alignment can be seen as breaking each of s_i and t_j into two parts and aligning corresponding parts of s_i and t_j as follows.

s_i	$(bab)^{p-1}$	$\mathbf{baa}(bab)^{q-p-1}(aab)(bab)^{n-q}$
t_j	$(aba)^{p-1}\mathbf{aba}$	$(aba)^{n-p}$

Similar to that in Case (1), the total score of this alignment is at least $n+7$. □

Corollary 1 *If the score for the constrained center-star sequence alignment is at most $m(n+3)$, the center sequence must be one of t_j for $1 \leq j \leq n$.*

Proof By Lemma 1, the alignment score between s_i and t_j for any i and j is at least $n+3$. It is obvious that the alignment score between s_i and s_j for any $i \neq j$ is more than 0. Thus, if s_i for some i is the center sequence, the total alignment score must be greater than $m(n+3)$. Therefore, we can assume that the center sequence must be one of t_j for $1 \leq j \leq n$. □

Lemma 2 *There is an independent set for G of size k if and only if there is a constrained center sequence alignment for the transformed instance $\Pi(G, k)$ of CCSA with score $m(n+3)$.*

Proof First, if G has an independent set Φ of size k, we give a constrained center sequence alignment with score $m(n+3)$. The alignment has t_1 (or any one of t_j for $1 \leq j \leq m$) as the center sequence. We have $t_j' = \text{-}(aba)^n\text{-}$ for $1 \leq j \leq n$. For a sequence s_i corresponding to edge $e_i = (v_p, v_q)$ with $p < q$, we have s_i' as follows.

- If $v_p \in \Phi$, $s_i' = \text{-}s_i = \text{-}(bab)^{p-1}(baa)(bab)^{q-p-1}(aab)(bab)^{n-q}$,
- If $v_p \notin \Phi$, $s_i' = s_i\text{-} = (bab)^{p-1}(baa)(bab)^{q-p-1}(aab)(bab)^{n-q}\text{-}$.

Recall that the constrained sequence is b^k. We prove that the constrain is satisfied in the alignment. Consider a vertex $v_x \in \Phi$. We can see that $t'_j[3x] = b$ for $1 \leq j \leq m$. For an edge $e_i = (v_x, v_y)$ incident to v_x and if $x < y$, then $s'_i == -(bab)^{x-1}(baa)(bab)^{y-x-1}(aab)(bab)^{n-y}$ and $s'_i[3x] = b$. If $y < x$, since $v_y \notin \Phi$, $s'_i == (bab)^{y-1}(baa)(bab)^{x-y-1}(aab)(bab)^{n-x}-$ and we also have $s'_i[3x] = b$. If an edge e_i is not incident to v_x, we have $s'_i[3x] = b$. Therefore, the constrain is satisfied.

For the alignment score, by Lemma 1, we can see that the score of the alignment t'_1 and s'_i for each $1 \leq i \leq m$ is $n + 3$ and the score of the alignment t'_1 and t'_j for each $1 \leq j \leq m$ is 0. Thus the total score is $m(n + 3)$.

Second, if there is a constrained center sequence alignment with score $m(n+3)$, we prove that there is an independent set Φ for G of size k. We prove it by contradiction. Assume that the maximum independent set of G is of size $k' < k$.

By Corollary 1, we can assume that t_1 is the center sequence. By Lemma 1, we can further assume that the pair-wise alignment score between t_1 and each of s_i for $1 \leq i \leq n$ is exactly $n + 3$ and the alignment follows either Alignment (1) or (2). As the constrain b^k is satisfied, we assume the xth "b" in b^k appear in column ℓ_x of the alignment matrix, in which the whole column should consist of "b" only. Suppose that column ℓ_x intersects with the $h(x)$th "aba" of t_1, and $h(x)$th "bab"/"baa"/"aab" of s_i for $1 \leq i \leq n$. We define a subset of vertex Φ that includes all $v_{h(x)}$ for $1 \leq x \leq k$.

We prove that Φ is an independent set of G. For any two vertices $v_{h(x)}$ and $v_{h(y)}$ in Φ for $x < y$, we claim that edge $e_i = (v_{h(x)}, v_{h(y)})$ does not exist. If not, there is a sequence $s_i = (bab)^{h(x)-1}(baa)(bab)^{h(y)-h(x)-1}(aab)(bab)^{n-h(y)}$. It can be verified that to follow Alignment (1) or (2), there is no way to have both the "b" in "baa" and the "b" in "aab" of s_i appear in column ℓ_x and column ℓ_y of the alignment matrix, respectively. Thus s_i, as well as e_i, does not exist. Therefore, Φ is an independent set of G and it is of size $k > k'$, which is a contradiction. □

By Lemma 2, we can reduce the NP-hard problem of MIS to CCSA with binary alphabet, and hence we have the following theorem.

Theorem 3 *Constrained Center-Star Sequence Alignment (CCSA) with binary alphabet is NP-Hard.*

3.2 Inapproximability of CCSA

In this part, we show that unless $P = NP$, no polynomial-time algorithm can solve the CCSA problem of an arbitrary distance function within a constant approximation ratio $r > 0$. If there is such an algorithm, we show that the MIS problem can be solved in polynomial time. In particular, we will focus on the CCSA problems where the distance function δ does not satisfy the triangle inequality.

We show a new reduction from MIS to CCSA which is similar to that in Sect. 3.1. The new transformation $\Pi'(G, k) = (\Sigma, S, c, \delta)$ is defined as follows. The alphabet set consists of one more character than before, i.e., $\Sigma = \{a, b, c\}$. The set of sequence $S = \{t_1, \ldots, t_m, s_1, \ldots, s_m\}$, where $t_1 = \cdots = t_m = c(aba)^n c$ and for each edge $e_i = (v_p, v_q)$ with $p < q$ $s_i = c(bab)^{p-1}(baa)(bab)^{q-p-1}(aab)(bab)^{n-q}c$. The constrained sequence is still $C = b^k$. The new distance function δ between any two characters in $\Sigma \cup \{-\}$ is defined as follows.

	a	b	c	–
a	0	1	2	B
b	1	0	2	B
c	2	2	0	0
–	B	B	0	0

where $B = r \cdot m(n+3)+1$. Note that the distance function may not satisfy the triangle inequality.

Similar to Lemma 2, we can show the new transformation Π' yields a polynomial-time reduction from MIS to CCSA.

Lemma 3 *There is an independent set for G of size k if and only if there is a constrained center sequence alignment for the transformed instance* $\Pi'(G, k)$ *of CCSA with score* $m(n+3)$.

In fact, for the transformed instance $\Pi'(G, k)$, one can obtain the optimal alignment if the guaranteed score is at most $r \cdot m(n+3)$.

Lemma 4 *If there is a constrained center sequence alignment for the transformed instance* $\Pi'(G, k)$ *of CCSA with score at most* $r \cdot m(n+3)$*, then this alignment has score exactly* $m(n+3)$.

Proof Consider the case that when there is a center constrained sequence alignment of $\Pi'(G, k)$ with score at most $r \cdot m(n+3)$. Since the distance between space character and a or b is greater than $r \cdot m(n+3)$, the alignment between s_i and t_1 (ignoring the "c") for any i must follow Alignment (1) or (2). Hence, the score of each pair-wise alignment is exactly $n+3$, and thus the total alignment score is $m(n+3)$. □

As a result, if there is an algorithm that gives a solution of $\Pi'(G, k)$ with approximation ratio r, then we can determine if an independent set for G of size k exists or not. If the algorithm gives an alignment of score at most $r \cdot m(n+3)$, then by Lemma 4 the optimal alignment has score $m(n+3)$ and then by Lemma 3, there is an independent set for G of size k. If the algorithm gives an alignment of score greater than $r \cdot m(n+3)$, then the optimal alignment has score greater than $m(n+3)$ and then by Lemma 3, there is no independent set for G of size k. Therefore, we have the following theorem for the inapproximability of CCSA.

Theorem 4 *Unless* $P = NP$*, there is no polynomial-time algorithm for CCSA with constant approximation ratio.*

4 Conclusion

We have studied the CPSA problem when the constrained sequence is a string of k identical characters and gave an $O(n^2)$ algorithm when $k = t$, where t is the minimum number of occurrences of that character in these two sequences, i.e, the largest number of that character in the constrained sequence. However, for some $k < t$ and problem instances, this CPSA problem might take $O(n^3)$ time. It is not sure whether this CPSA problem can be solved in strictly less than $O(n^3)$ time for all k.

Two negative results of the CCSA problem were shown in this paper. However, whether there exist constant approximation algorithms for binary alphabets or for 3-letter alphabets with distance function satisfying triangle inequality are both interesting open problems.

Since the solution for the center-star alignment problem can provide a good approximation for the MSA problem, the solution for the CCSA problem can also give a good approximation for the CMSA problem. Unfortunately, the CCSA problem has been proved to be NP-complete and also difficult to find an approximate solution for some distance function. Thus, it remains open whether there exists a good approximation algorithm for the CMSA problem.

Acknowledgments The authors thank the anonymous referees for their helpful comments to improve the presentation of this paper. This work was supported by NSFC (61433012, U1435215, 11171086), HK RGC Grant (HKU 7114/13E, HKU 7164/12E, HKU 7111/12E), HKU small project funding 201309176064, Natural Science Foundation of Hebei A2013201218, Chinese Academy of Sciences research Grant (No. KGZD-EW-103-5(9)), Fundamental Research Foundation of Northwestern Polytechnical University in China (Grant No. JC201164), Fundamental Research Funds for the Central Universities (Grant No. 3102015ZY081), and China Postdoctoral Science Foundation (Grant No. 2012M521803).

References

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. J Mol Biol 215(3):403–410

Bonizzoni P, Vedova GD (2001) The complexity of multiple sequence alignment with sp-score that is a metric. Theor Comput Sci 259(1–2):63–79

Chin FYL, Santis AD, Ferrara AL, Ho NL, Kim SK (2004) A simple algorithm for the constrained sequence problems. Inf Process Lett 90:175–179

Chin FYL, Ho NL, Lam TW, Wong PWH (2005) Efficient constrained multiple sequence alignment with performance guarantee. J Bioinform Comput Biol 3(1):1–18

Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) Introduction to algorithms, 3rd edn. The MIT Press, Cambridge

Garey M, Johnson D (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman and Company, San Francisco

Gusfield D (1993) Efficient methods for multiple sequence alignment with guaranteed error bounds. Bul Math Biol 55:141–154

Iliopoulos CS, Rahman MS (2008) Algorithms for computing variants of the longest common subsequence problem. Theor Comput Sci 395(2–3):255–267

Larkin MA, Blackshields G, Brown NP, Chenna R, McGettigan PA, McWilliam H, Valentin F, Wallace IM, Wilm A, Lopez R, Thompson JD, Gibson TJ, Higgins DG (2007) ClustalW and ClustalX version 2. Bioinformatics 23(21):2947–2948

Masek WJ, Paterson MS (1980) A faster algorithm computing string edit distances. J Comput Syst Sci 20(1):18–31

Mount DM (2004) Bioinformatics: sequence and genome analysis, 2nd edn. Cold Spring Harbor Laboratory Press, Cold Spring Harbor

Setubal J, Meidanis J (1997) Introduction to computational molecular biology (Chap. 3). PWS Publishing Company, Boston

Tang CY, Lu CL, Chang MD-T, Tsai Y-T, Sun Y-J, Chao K-M, Chang J-M, Chiou Y-H, Wu C-M, Chang H-T, Chou W-I (2003) Constrained multiple sequence alignment tool development and its application to rnase family alignment. J Bioinform Comput Biol 1(2):267–287

Wang L, Jiang T (1994) On the complexity of multiple sequence alignment. J Comput Biol 1(4):337–348