



The Maximum Duo-Preservation String Mapping Problem with Bounded Alphabet*

Nicolas Boria  

Université Paris-Dauphine, Université PSL, CNRS, LAMSADE, 75016, Paris, France

Laurent Gourvès 

Université Paris-Dauphine, Université PSL, CNRS, LAMSADE, 75016, Paris, France

Vangelis Th. Paschos  

Université Paris-Dauphine, Université PSL, CNRS, LAMSADE, 75016, Paris, France

Jérôme Monnot  

Université Paris-Dauphine, Université PSL, CNRS, LAMSADE, 75016, Paris, France

Abstract

Given two strings A and B such that B is a permutation of A , the MAX DUO-PRESERVATION STRING MAPPING (MPSM) problem asks to find a mapping π between them so as to preserve a maximum number of duos. A duo is any pair of consecutive characters in a string and it is preserved by π if its two consecutive characters in A are mapped to same two consecutive characters in B . This problem has received a growing attention in recent years, partly as an alternative way to produce approximation algorithms for its minimization counterpart, MIN COMMON STRING PARTITION, a widely studied problem due its applications in comparative genomics. Considering this favored field of application with short alphabet, it is surprising that MPSM^ℓ , the variant of MPSM with bounded alphabet, has received so little attention, with a single yet impressive work that provides a 2.67-approximation achieved in $O(n)$ [5], where $n = |A| = |B|$. Our work focuses on MPSM^ℓ , and our main contribution is the demonstration that this problem admits a Polynomial Time Approximation Scheme (PTAS) when $\ell = O(1)$. We also provide an alternate, somewhat simpler, proof of NP-hardness for this problem compared with the NP-hardness proof presented in [16].

2012 ACM Subject Classification Theory of computation \rightarrow Approximation algorithms analysis; Theory of computation \rightarrow Dynamic programming; Theory of computation \rightarrow Pattern matching; Theory of computation \rightarrow Complexity classes

Keywords and phrases Maximum-Duo Preservation String Mapping, Bounded alphabet, Polynomial Time Approximation Scheme

Digital Object Identifier 10.4230/LIPIcs.WABI.2021.5

Funding *Nicolas Boria*: Supported by Agence Nationale de la Recherche (ANR), project STAP ANR-17-CE23-0021.

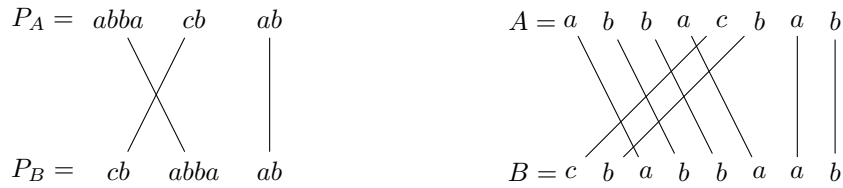
Laurent Gourvès: Supported by Agence Nationale de la Recherche (ANR), project THEMIS ANR-20-CE23-0018

1 Introduction

Evaluating the similarity between strings is a core problem of stringology, with various applications ranging from data compression to bioinformatics. Various distance measures on string have been proposed such as the Hamming distance (see [23] for a full survey), the Jaro-Winkler distance [28], the overlap coefficient [27], etc. However, in real life applications, many of these simple distance measures fail to provide significant information, and the

* This work is dedicated to the memory of our colleague and friend Jérôme Monnot who untimely passed away.





(a) A solution expressed as blocks. (b) The same solution expressed as mapping.

■ **Figure 1** An instance of MCSP along with its solution.

computation of more involved distances is often required. One of the most commonly used distance measure between two strings is the so called *edit distance* [18]. Given a set of allowed operations on strings, and a cost function on the allowed operations, the edit distance measures the minimum overall cost of edit operations that need to be performed to transform the first string into the second. The edit distance is a versatile concept that applies to various complex objects apart from strings, and allows for the representation of many different problems by considering different sets of allowed edit operations and different cost functions. In data compression, the edit distance can be used to help store efficiently a couple or a set of similar yet different data (e.g., different versions of the same object), by encoding a base element only, and then representing each other element of the dataset as the series of edit operations that results in it starting from the base element [25]. In bioinformatics, the edit distance provides some measure of kinship between species by measuring the similarity of their DNA [17, 24].

When the only allowed edit operation is shifting a block of characters within the string, computing the edit distance between two strings amounts to solving MIN COMMON STRING PARTITION problem. The MIN COMMON STRING PARTITION (MCSP) is a fundamental problem in the field of string comparison [9, 15], and can be applied more specifically to genome rearrangement problems, as shown in [26], where each block of the partition can be seen as a gene. Consider two strings A and B , both of length n , such that B is a permutation of A . MCSP asks for a partition \mathcal{P}_A of A and a partition \mathcal{P}_B of B , both of minimum cardinality $|\mathcal{P}_A| = |\mathcal{P}_B| = p$, such that \mathcal{P}_A is a permutation of \mathcal{P}_B . As B can be reconstructed by shifting $p - 1$ blocks of \mathcal{P}_A , the edit distance between A and B is equal to $p - 1$. Figure 1 provides a visual representation of the problem with $A = abbacbab$ and $B = cbabbaab$.

By focusing on how the blocks of \mathcal{P} are mapped between A and B , the problem can be alternatively defined as follows: given two strings A and B , both of length n , such that B is a permutation of A , one needs to define a mapping π that maps each position i of A to a position $\pi(i)$ of B such that, for all $i \in \{1, \dots, n\}$, the letter $A[i]$ is the same as the letter $B[\pi(i)]$. The number $p - 1$ of cuts in a partition with p blocks is then equal to the number of consecutive positions, or *duos*, $(i, i + 1)$ such that $\pi(i) + 1 \neq \pi(i + 1)$ (in Figure 1b, this happens for $i = 4$ and $i = 6$). Hence, maximizing the number of duos that are *preserved* by the mapping, i.e., such that $\pi(i) + 1 = \pi(i + 1)$ immediately yields a solution for MCSP. This gives rise to the maximization version of the MSCP problem, called MAX DUO-PRESERVATION STRING MAPPING (MPSM):

► **Definition 1.** MAX DUO-PRESERVATION STRING MAPPING

Given two strings A and B of length n such that A is a permutation of B , the MAX DUO-PRESERVATION STRING MAPPING problem (MPSM) asks for a bijective mapping π between positions of A and B such that:

- $\forall i \in \{1, \dots, n\}, A[i] = B[\pi(i)]$
- the number of duos preserved by π is maximum

Related works

k -MCSP denotes the restricted version of MCSP where each character occurs at most k times. This problem has been shown NP-hard and APX-hard even with $k = 2$ [15]. Several approximations are known for low values of k [9, 10, 11, 15, 19, 20], and the problem was also studied extensively in terms of parameterized algorithms [6, 7, 12, 16]. Another variant of MCSP, denoted by MCSP^ℓ , deals with the version where the alphabet used to form the strings consists of at most ℓ characters. This version of the problem has been proved to be NP-hard [16] for any $\ell \geq 2$. The best approximation ratio known so far for the general version is $O(\log n \log^* n)$ [11].

In order to tackle the approximation issue from a different angle, the maximization version of the problem (as described in Definition 1) was introduced in [8], with an $O(k^2)$ approximation for k -MPSM (where, similarly to k -MCSP, each character occurs at most k times). The problem has been shown APX-hard in [3] even with $k = 2$, and the article also provided the first constant approximation for the general problem with a simple 4-approximation algorithm. This ratio was later improved to 3.5 using a local search technique [2], 3.25 through a combinatorial triplet matching approach [4], and finally $2 + \varepsilon$ in $n^{O(1/\varepsilon)}$ with a combination of a greedy algorithm and local search [13]. Note that [13] also presented a 2.67-approximation algorithm with time complexity $O(n^2)$. Moreover a $1.4 + \varepsilon$ approximation algorithm for 2-MCSP appears in [29]. Regarding the version of MCSP where the alphabet has at most ℓ characters (MCSP^ℓ), a recent work proposed a 2.67-approximation algorithm that requires time as low as $O(n + \ell^7)$ [5]. The method presented in [5] also guarantees a 2.67-approximation in $O(n^3)$ for the weighted version of the problem which was introduced in [22]. The weighted version takes into consideration the positions of the preserved duos (the closer the better). Finally, the problem was also studied through the prism of fixed-parameter tractability, and was shown to be FPT with respect to the number of preserved duos in [1], whereas [21] presented more efficient algorithms still parameterized with respect to the solution size.

Our Contribution

In this work, we tackle the MCSP^ℓ problem where the alphabet used to form strings A and B has at most $\ell = O(1)$ characters. The problem is NP-hard even with $\ell = 2$, as its minimization counterpart has been shown to be NP-hard in [16]. We do however provide a more direct reduction in Section 4. The main contribution of this article consists of the proof that MCSP^ℓ admits a Polynomial Time Approximation Scheme (PTAS), as we provide an algorithm based on dynamic programming that guarantees, for every fixed integer $k > 1$, a $(1 + \frac{1}{k-1})$ -approximation within time complexity $O(kn^{1+c\ell^k})$, where c is some constant, which amounts to $O(n^{O(1)})$ provided that both k and ℓ are constant. The algorithm and its approximation analysis are presented in Section 3.

We remind that a PTAS is a family of approximation algorithms. For any fixed $\varepsilon > 0$, a PTAS for a maximization problem runs in time polynomial in the instance size and outputs a feasible solution of value SOL such that $(1 + \varepsilon)SOL \geq OPT$ where OPT denotes the optimal value achieved by a feasible solution. A *fully* polynomial time approximation scheme (FPTAS) also satisfies $(1 + \varepsilon)SOL \geq OPT$ but its running time is polynomial both in the instance size and $1/\varepsilon$.

Since the objective value of MCSP^ℓ is upper bounded by a polynomial of the instance size (indeed, at most $n - 1$ duos can be preserved), the existence of an FPTAS for MCSP^ℓ would lead to the unlikely fact that $P=NP$.

2 Preliminaries

Let A and B represent two strings formed on an alphabet \mathcal{L} , such that B is a permutation of A . Let $n = |A| = |B|$. We denote by ℓ the size of the alphabet \mathcal{L} .

A *word* w on alphabet \mathcal{L} is a non empty tuple of letters from \mathcal{L} . The i -th letter of w is denoted by $w[i]$ (similar to the fact that the i -th letter of A is denoted by $A[i]$). $\mathcal{D}_k(\mathcal{L})$ is the complete dictionary of words on \mathcal{L} with at most k letters (i.e., the word size is at most k). Note that $|\mathcal{D}_k(\mathcal{L})| = \sum_{i=1}^k \ell^i = O(\ell^k)$. When the context is clear, the dictionary is denoted by \mathcal{D} .

Cardinality assignments X and $[w]$

In what follows, X denotes a vector in $\mathbb{N}^{|\mathcal{D}_k(\mathcal{L})|}$ called *cardinality assignment*, that assigns a cardinality¹ to every word of $\mathcal{D}_k(\mathcal{L})$. $X(w)$ denotes the cardinality assigned to word w in X . $[w]$ denotes the cardinality assignment that assigns 1 to w and 0 to every other word.

Regular operations on vectors are allowed on cardinality assignments. In particular, the cardinality assignment $X + [w]$ (resp., $X - [w]$) is identical to X except that position w is increased (resp., decreased) by one unit.

Finally, we denote by X_0 the cardinality assignment that assigns 0 to every word.

String description and String-cuts

We call *string-cut* a partitioning S of a string A in sub-strings. Concretely, a string-cut is an ordered list of words whose concatenation results in A . We say that a cardinality assignment X *describes* a string-cut S if words in S appear with the exact frequencies described by X . Similarly, we say that a cardinality assignment X *describes* a string A if there exists a string-cut S of A such that X describes S .

A string-cut is said to be *k-bounded* if its words have length at most k . We denote by S_t the t -th word of a string-cut S , while the operation $S :: w$ consists of appending word w to the end of S .

Example

Consider the string $A = \text{abbabbaab}$ formed on the alphabet $\mathcal{L} = \{a, b\}$. The dictionary $\mathcal{D}_2(\mathcal{L})$ contains all possible words of length at most two using letters of alphabet \mathcal{L} . Namely, $\mathcal{D}_2(\mathcal{L}) = \{a, b, aa, bb, ab, ba\}$. The string A can be partitioned in the following 2-bounded string-cut $S = (ab, b, ab, ba, ab)$ described by the cardinality assignment $X = (0, 1, 0, 0, 3, 1)$ (because S has zero a , one b , zero aa , zero bb , three ab , and one ba). Hence, it holds that the cardinality assignment X describes A .

Cardinality assignment evaluation

For any cardinality assignment X , we introduce the function $\text{eval}(X)$, which returns the number of duos of characters *within the words of the cardinality assignment*, namely:

$$\text{eval}(X) = \sum_{w \in \mathcal{D}} (|w| - 1) \cdot X(w).$$

¹ How many times the word occurs.

When two string-cuts S_A and S_B are described by the same cardinality assignment X (i.e., S_A is a permutation of S_B), any mapping between words of S_A of S_B translates in a mapping between letters of A and letters of B that preserves *at least* $\text{eval}(X)$ duos. There are possibly more preserved duos if consecutive words in S_A are mapped to consecutive words in S_B as shown in Figure 2, where two strings A and B are indeed represented by two string-cuts that are described by the same cardinality assignment. On the one hand, the mapping in Figure 2a preserves only duos that are within words of the string-cuts, and hence it saves 4 duos. On the other hand, the mapping in Figure 2b maps two consecutive words in S_A to two consecutive words in S_B so that an extra duo bb is preserved.



(a) A mapping that preserves $\text{eval}(X) = 4$ duos. (b) A mapping that preserves $\text{eval}(X) + 1 = 5$ duos.

■ **Figure 2** Two mappings between S_A and S_B with $A = \text{abbabbaab}$ and $B = \text{ababbabab}$.

Also, notice that given two string-cuts on A and B that are described by the same cardinality assignment X , a mapping between words of the string-cuts can be easily constructed: map the i -th occurrence of each word in S_A to its i -th occurrence in S_B . Since any mapping of words preserves at least $\text{eval}(X)$ duos, our method aims at computing two string cuts S_A and S_B that are both described by some cardinality assignment X , such that $\text{eval}(X)$ is a lower bound on the number of duos preserved by some mapping which may be reconstructed by arbitrarily mapping words of S_A to words of S_B . We now move on to the description of our algorithm.

3 A Polynomial Time Approximation Scheme for MPSM^ℓ

In the following, we describe a method based on dynamic programming which results in a polynomial time approximation scheme (PTAS) for MSPM^ℓ .

For the sake of clarity, the algorithm is presented in a two-fold fashion. In Section 3.1 we describe and analyze a procedure called $\text{GENERATE}(A, k)$ which, given a string A and an integer k , uses dynamic programming to generate an exhaustive collection of possible cardinality assignments that describe some k -bounded string-cut of A . One important feature that ensures the polynomial complexity of our method is the following: although the number of different k -bounded string-cuts of A and B grows exponentially with n , the number of different cardinality assignments describing these string-cuts is upper bounded by a polynomial in n provided that both k and ℓ are constant. In Section 3.2, we show how to use the data structure created by the procedure $\text{GENERATE}(A, k)$ to find the best possible cardinality assignment that describes both A and B , and to generate a matching between characters of A and B that saves the maximum number of duos within k -bounded string-cuts of A and B . Finally, in Section 3.3, we show how the method yields a PTAS for MSPM^ℓ .

3.1 Algorithm GENERATE(A, k)

■ **Algorithm 1** GENERATE(A, k).

Require: a string A and a positive integer k

Ensure: a couple (Ω, \mathcal{S}) such that

- Ω is a set of cardinality assignments such that any k -bounded string-cut of A is described by some cardinality assignment $X \in \Omega$
- \mathcal{S} is a dictionary that assigns a single string-cut $\mathcal{S}(X)$ to every cardinality assignment $X \in \Omega$

```

1:  $\Omega_0 \leftarrow \{X_0\}$ 
2:  $\mathcal{S}(X_0) = \emptyset$ 
3: for  $i = 1$  to  $n$  do
4:    $\Omega_i \leftarrow \emptyset$ 
5:   for each  $j \in [1, \min(i, k)]$  do
6:      $w \leftarrow (A[i - j + 1], \dots, A[i])$ 
7:     for each  $X \in \Omega_{i-j}$  do
8:       if  $(X + [w]) \notin \Omega_i$  then
9:          $\Omega_i \leftarrow \Omega_i \cup (X + [w])$ 
10:         $\mathcal{S}(X + [w]) \leftarrow \mathcal{S}(X) :: w$ 
11:       end if
12:     end for
13:   end for
14: end for
15:  $\Omega \leftarrow \Omega_n$ 
16: return  $(\Omega, \mathcal{S})$ 

```

Algorithm 1 is based on dynamic programming and generates a collection of cardinality assignments Ω , as well as a dictionary \mathcal{S} of corresponding string-cuts, such that for each $X \in \Omega$, there exists a single string-cut $\mathcal{S}(X) \in \mathcal{S}$ that describes X . When building Ω_i , the algorithm considers words w of length $j \leq k$ which are sub-strings of A and whose last character is $A[i]$. Then, it appends w onto a string-cut $\mathcal{S}(X)$ such that $X \in \Omega_{i-j}$, and store the resulting string-cut in \mathcal{S} .

Let us describe the data structures that are created throughout Algorithm 1. GENERATE(A, k) aims at computing:

- a complete collection Ω containing all possible cardinality assignments describing some k -bounded string-cut of A ,
- a corresponding dictionary of string-cuts \mathcal{S} , such that for all $X \in \Omega$, there exists a unique string-cut in \mathcal{S} described by X . $\mathcal{S}(X)$ denotes the unique string-cut in \mathcal{S} described by X .

The first property of (Ω, \mathcal{S}) is proved in the following (Proposition 2). The second property is ensured by Lines 8-10: a single entry is added to the dictionary \mathcal{S} (Line 10) only when a new cardinality assignment is added to some set Ω_i (Line 9), while Line 8 ensures that there is no duplicate entries within the Ω_i 's.

Let us now prove that the collection Ω produced by Algorithm 1 contains all the cardinality assignments necessary to describe any k -bounded string-cut of A .

► **Proposition 2.** *For any k -bounded string-cut S of A , it holds that there exists $X \in \Omega$ that describes S .*

Proof. We demonstrate the following claim by induction on i .

▷ **Claim 3.** Any k -bounded string-cut S of the sub-string $A_i = (A[1], \dots, A[i])$ is described by some cardinality assignment X of Ω_i .

The claim holds trivially for $i = 0$ and $i = 1$, as one can verify that after the first iteration, we have $\Omega_1 = \{A[1]\}$. In other words, Ω_1 contains a single cardinality assignment, the one that describes a single word with a single letter (the first letter of A).

Now, let us suppose that Claim 3 is verified for all $j < i$ and prove that it is verified also for i .

Let S be a k -bounded string-cut of A_i , let X' denote the cardinality assignment that describes S , and let \tilde{w} with length $|\tilde{w}|$ denote the last word of S . We need to prove that X' belongs to Ω_i by the end of the algorithm.

By hypothesis, the claim holds for $j = i - |\tilde{w}|$. In other words, there is a cardinality assignment $\tilde{X} \in \Omega_{i-|\tilde{w}|}$ that describes the string-cut $S \setminus \tilde{w}$ of $A_{i-|\tilde{w}|}$. Moreover, \tilde{X} is equal to $X' - [\tilde{w}]$ as the string cut \tilde{X} corresponds to the string-cut X' with one less occurrence of word \tilde{w} .

Consider the i^{th} iteration of the outer loop starting at Line 3 of Algorithm 1 and the inner loop (Line 5) where $j = |\tilde{w}|$ (Recall that S is k -bounded so $|\tilde{w}| \leq k$). By induction hypothesis, it holds that \tilde{X} belongs to $\Omega_{i-|\tilde{w}|}$, so there will be an iteration of the innermost loop (Line 7) with $X = \tilde{X}$: at this point of the algorithm, it will be checked if the cardinality assignment $\tilde{X} + [w] = X'$ already belongs to Ω_i , and will include it to Ω_i if it is not the case.

Hence, by the end of the algorithm, it holds that $X' \in \Omega_i$, and the proof is complete. ◀

Regarding the complexity of Algorithm 1, let us stress that no set Ω_i has any duplicate entries (ensured by the *if* condition at Line 8), so that its cardinality is bounded by the number of different possible cardinality assignments describing any k -bounded string-cut over a string of length i . Recall that such cardinality assignment is a vector with $|\mathcal{D}_k(\mathcal{L})|$ coordinates, each coordinate describing the frequency of a word in a string-cut. Roughly, each coordinate of a cardinality assignment of Ω_i is upper bounded by i , so that there are at most $i^{|\mathcal{D}_k(\mathcal{L})|}$ different cardinality assignments, i.e., $|\Omega_i| \leq i^{|\mathcal{D}_k(\mathcal{L})|}$. Moreover, let us remind that there exists a constant c such that $|\mathcal{D}_k(\mathcal{L})| \leq c \cdot |\mathcal{L}|^k$, and thus $|\Omega_i| \leq i^{c \cdot |\mathcal{L}|^k}$.

Hence, the total number of innermost loops that are made at the i^{th} iteration of the outer loop is at most $k|\Omega_{i-1}|$. The overall complexity is thus $\sum_{i=1}^n k|\Omega_i|$ which is bounded by $k \sum_{i=1}^n i^{c \cdot |\mathcal{L}|^k} = O(n \cdot kn^{c \cdot |\mathcal{L}|^k}) = O(kn^{c \cdot |\mathcal{L}|^k + 1})$.

3.2 Algorithm MATCH(A, B, k)

We now devise the following algorithm, called MATCH (see Algorithm 2), parameterized by k for MPSM, which first runs Algorithm 1 on A and B in order to compute the couples $(\Omega_A, \mathcal{S}_A)$ and $(\Omega_B, \mathcal{S}_B)$.

Algorithm 2 consists of identifying the cardinality assignment X^{SOI} that belongs to both Ω_A and Ω_B and which contains the largest number of duos. Along with the identification of X^{SOI} comes the identification of two string-cuts (namely, S_A^{SOI} and S_B^{SOI} for A and B , respectively) which are both described by X^{SOI} .

The complexity of the loop in MATCH- $\Omega(k)$ can be brought down to $O(|\Omega_A|)$ with a proper data structure, that is, a structure that ensures a fast (linear time) verification of the condition expressed at Line 5. Hence the overall complexity is given by the generation of couples $(\Omega_A, \mathcal{S}_A)$ and $(\Omega_B, \mathcal{S}_B)$, namely $O(kn^{1+c \cdot |\mathcal{L}|^k})$.

Algorithm 2 MATCH(A, B, k).

```

1: Initialize  $S_A^{SOL} = \emptyset, S_B^{SOL} = \emptyset, X^{SOL} = X_0$ 
2:  $(\Omega_A, \mathcal{S}_A) \leftarrow \text{GENERATE}(A, k)$ 
3:  $(\Omega_B, \mathcal{S}_B) \leftarrow \text{GENERATE}(B, k)$ 
4: for each  $X \in \Omega_A$  do
5:   if  $X \in \Omega_B$  and  $\text{eval}(X) > \text{eval}(X^{SOL})$  then
6:      $X^{SOL} \leftarrow X$ 
7:   end if
8: end for
9:  $S_A^{SOL} \leftarrow \mathcal{S}_A(X^{SOL})$ 
10:  $S_B^{SOL} \leftarrow \mathcal{S}_B(X^{SOL})$ 
11: return  $(S_A^{SOL}, S_B^{SOL})$ 

```

As mentioned at the end of Section 2, a mapping between letters of A of B that preserves at least $\text{eval}(X^{SOL})$ duos can be easily derived from the couple (S_A^{SOL}, S_B^{SOL}) computed by Algorithm 2. Now it remains to make the link between $\text{eval}(X^{SOL})$ and the number of duos preserved by an optimal solution.

3.3 Approximation Analysis

We now prove that MATCH(A, B, k) yields a Polynomial Time Approximation Scheme for MPSM ^{ℓ} .

First, we need to provide some lower bound on the number of duos that are preserved by our solution.

► **Proposition 4.** *Let S_A^{SOL} and S_B^{SOL} be the string-cuts generated by MATCH(A, B, k). Both are described by the same cardinality assignment X^{SOL} .*

Given any couple of k -bounded string-cuts \tilde{S}_A on A and \tilde{S}_B on B that are both described by the same cardinality assignment \tilde{X} , it holds that $\text{eval}(X^{SOL}) \geq \text{eval}(\tilde{X})$.

Proof. From Proposition 2, we can assert that $\tilde{X} \in \Omega_A$ and $\tilde{X} \in \Omega_B$. Hence, in the first part of the MATCH(A, B, k), the cardinality assignment \tilde{X} will be considered as a candidate. Eventually, the algorithm retains X^{SOL} that yields the best solution among all candidates considered, including \tilde{X} . ◀

Consider now an optimal solution π^* for MPSM ^{ℓ} that preserves OPT duos, and an approximate solution π returned by Algorithm 2 that preserves $SOL \geq \text{eval}(X^{SOL})$ duos. As stated in Section 2, π^* induces string-cuts S_A^* and S_B^* of A and B , such that S_A^* is a permutation of S_B^* . These string-cuts may contain words that are strictly longer than k . Consider the string-cuts S'_A and S'_B that result from cutting all such words into slices of length at most k in both S_A^* and S_B^* , such that S'_A and S'_B are both described by the same cardinality assignment, say X' . No more than OPT/k additional cuts are added this way. It holds that:

- S'_A and S'_B are k -bounded, and they are described by the same cardinality assignment X' . Hence, by applying Proposition 2, it holds that $SOL \geq \text{eval}(X')$.
- The optimal mapping π^* preserves $\text{eval}(X')$ duos within words of X' and at most OPT/k duos between words of X' , one for each added cut. Hence, it holds that $OPT \leq \text{eval}(X') + OPT/k$.

Combining these two facts, one immediately derives the following approximation between OPT and SOL :

$$\frac{OPT}{SOL} \leq 1 + \frac{1}{k-1}, \quad \forall k > 1.$$

By fixing $k = \lceil 1/\varepsilon + 1 \rceil$, we can then assert that our algorithm guarantees a $(1 + \varepsilon)$ -approximation within time complexity $O(n^{O(1)})$ provided that both k and ℓ are bounded by some constant. Hence, our final result holds:

► **Proposition 5.** *MPSM $^\ell$ with $\ell = O(1)$ admits a PTAS.*

4 Hardness result

The fact that MCSP² is NP-complete is known from [16, Theorem 1]. However, we give an alternate, somewhat simpler, proof.

► **Proposition 6 (Alternate Proof).** *MPSM $^\ell$ and MCSP $^\ell$ are both NP-hard, even if $\ell = 2$.*

Proof. Consider an instance I of the NP-complete problem 3-PARTITION [14], with a set of integers $X = \{x_1, \dots, x_n\}$ such that $n = 3m$, and let $m\Sigma = \sum_{x_i \in X} x_i$. As standard hypothesis for 3-PARTITION, suppose $\Sigma/4 < x_i < \Sigma/2$ holds for all x_i . The question is whether X can be partitioned in m triplets, each of total sum Σ .

We build an instance $J(I)$ of MPSM $^\ell$ with $\ell = 2$, consisting of two strings A and B built in the following way.

- For each integer x_i in X , we build a string A_i that consists of one b followed by $x_i + 1$ consecutive a 's. We also build a sub-string A_{n+1} that contains a single letter b . String A is the concatenation of all strings A_i 's.
- Consider the substring B_i that consists of one b followed by $\Sigma + 3$ letters a , followed by 2 letters b . String B results from the concatenation of m substrings B_i , with a single additional b at the very start. Note that all B_i 's are identical, we merely index them by i to simplify the proof.

One can easily verify that both strings A and B contain exactly $m\Sigma + n$ occurrences of letter a and $n + 1$ occurrences of letter b , so $J(I)$ is a valid instance of MCSP².

We are going to show that instance I of 3-PARTITION admits a solution if and only if instance $J(I)$ of MPSM² admits a solution that preserves $m\Sigma + 2m$ duos.

(\Rightarrow) First, note that in string A , there are exactly n ba duos, n ab duos, $m\Sigma$ aa duos, and no bb duo. On the other hand, string B has m ab duos, m ba duos, $m\Sigma + 2m$ aa duos, and $2m$ bb duos.

Thus, any solution will preserve at most $m\Sigma + 2m$ duos in total, that is, $\min\{m\Sigma, m\Sigma + 2m\} = m\Sigma$ duos of type aa , $\min\{m, n\} = m$ duos of type ab , $\min\{0, 2m\} = 0$ duos of type bb , and $\min\{m, n\} = m$ duos of type ba .

Suppose I is a yes-instance of 3-PARTITION. A solution is given by a set of triplets $\{T_1, \dots, T_m\}$. Using this set, we can construct a solution to $J(I)$ which preserves exactly $m\Sigma + 2m$ duos.

For each $T_i = \{x_{i1}, x_{i2}, x_{i3}\}$, with $x_{i1} \leq x_{i2} \leq x_{i3}$, we map the whole sub-string A_{i1} to the first $x_{i1} + 2$ characters of sub-string B_i , we map all a 's of A_{i2} to the following $x_{i2} + 1$ a 's of B_i , and finally we map all a 's of A_{i3} and the first letter of A_{i3+1} to the remaining part of B_i except for its final b , which remains unmatched for now.

The unmatched b 's of A are arbitrarily matched with the remaining b 's of B (their quantities are equal because B is a permutation of A).

The resulting overall mapping preserves every duo aa of A and all duos ab and ba of B , that is, a total of $m\Sigma + 2m$ duos. Thus, the first part of the reduction is complete.

(\Leftarrow) Now suppose that there exists a mapping between A and B which preserves $m\Sigma + 2m$ duos. Since A and B have exactly $m\Sigma + 2m$ duos in common, the mapping must preserve all aa duos of A and all duos of type ab and ba of B .

Since all the aa duos of A are preserved, letters a of the word associated with any number of X cannot be mapped with letters a of more than one B_i . Since each B_i has exactly $m\Sigma + 3$ letters a , each B_i receives words associated with numbers of X whose sum is at most $m\Sigma$. If one B_i receives words associated with numbers whose sum is strictly less than $m\Sigma$, then not all the aa duos of A are preserved, leading to a contradiction. Therefore, each B_i hosts a set of numbers whose sum is Σ . In other words, a 3-partition of the numbers is derived. \blacktriangleleft

5 Conclusion

When the alphabet used to form the instance is bounded, an exhaustive list of all possible cardinality assignments describing k -bounded string-cuts can be produced in polynomial time for any constant k . This fact helped us devise a Polynomial Time Approximation Scheme for MPSM^ℓ , which is the best result one might expect as no FTPAS can exist for this problem unless $\text{P}=\text{NP}$. Future developments include inquiries as to how the techniques presented in this article can be adapted to tackle the weighted version of MPSM introduced in [22]. Though polynomial, the time complexity of our method may well prove to be intractable in practical cases with large instances. That is why we also intend to devise different approaches for MPSM^ℓ and evaluate them through experiments.

References

- 1 Stefano Beretta, Mauro Castelli, and Riccardo Dondi. Parameterized tractability of the maximum-duo preservation string mapping problem. *Theoretical Computer Science*, 646:16–25, 2016.
- 2 Nicolas Boria, Gianpiero Cabodi, Paolo Camurati, Marco Palena, Paolo Pasini, and Stefano Quer. A $7/2$ -Approximation Algorithm for the Maximum Duo-Preservation String Mapping Problem. In *27th Annual Symposium on Combinatorial Pattern Matching (CPM 2016)*, pages 11:1–11:8, 2016.
- 3 Nicolas Boria, Adam Kurpisz, Samuli Leppänen, and Monaldo Mastrolilli. Improved approximation for the maximum duo-preservation string mapping problem. In *International Workshop on Algorithms in Bioinformatics (WABI 2014)*, pages 14–25. Springer, 2014.
- 4 Brian Brubach. Further improvement in approximating the maximum duo-preservation string mapping problem. In *International Workshop on Algorithms in Bioinformatics (WABI 2016)*, pages 52–64. Springer, 2016.
- 5 Brian Brubach. Fast matching-based approximations for maximum duo-preservation string mapping and its weighted variant. In *Annual Symposium on Combinatorial Pattern Matching (CPM 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
- 6 Laurent Bulteau, Guillaume Fertin, Christian Komusiewicz, and Irena Rusu. A fixed-parameter algorithm for minimum common string partition with few duplications. In *Algorithms in Bioinformatics - 13th International Workshop, (WABI 2013)*, pages 244–258, 2013.
- 7 Laurent Bulteau and Christian Komusiewicz. Minimum common string partition parameterized by partition size is fixed-parameter tractable. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 2014)*, pages 102–121, 2014.
- 8 Wenbin Chen, Zhengzhang Chen, Nagiza F. Samatova, Lingxi Peng, Jianxiong Wang, and Maobin Tang. Solving the maximum duo-preservation string mapping problem with linear programming. *Theoretical Computer Science*, 530:1–11, 2014.

- 9 Xin Chen, Jie Zheng, Zheng Fu, Peng Nan, Yang Zhong, Stefano Lonardi, and Tao Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE ACM Trans. Comput. Biol. Bioinform.*, 2(4):302–315, 2005.
- 10 Marek Chrobak, Petr Kolman, and Jiri Sgall. The greedy algorithm for the minimum common string partition problem. In *Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques, 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2004, and 8th International Workshop on Randomization and Computation, RANDOM 2004, Cambridge, MA, USA, August 22-24, 2004, Proceedings*, pages 84–95, 2004.
- 11 Graham Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. *ACM Trans. Algorithms*, 3(1):2:1–2:19, 2007.
- 12 Peter Damaschke. Minimum common string partition parameterized. In *Algorithms in Bioinformatics, 8th International Workshop, WABI 2008, Karlsruhe, Germany, September 15-19, 2008. Proceedings*, pages 87–98, 2008.
- 13 Bartłomiej Dudek, Pawel Gawrychowski, and Piotr Ostropolski-Nalewaja. A family of approximation algorithms for the maximum duo-preservation string mapping problem. In *28th Annual Symposium on Combinatorial Pattern Matching (CPM 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
- 14 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- 15 Avraham Goldstein, Petr Kolman, and Jie Zheng. Minimum common string partition problem: Hardness and approximations. In *Algorithms and Computation, 15th International Symposium, ISAAC 2004, Hong Kong, China, December 20-22, 2004, Proceedings*, pages 484–495, 2004.
- 16 Haitao Jiang, Binhai Zhu, Daming Zhu, and Hong Zhu. Minimum common string partition revisited. *J. Comb. Optim.*, 23(4):519–527, 2012.
- 17 Bruce Johnson. A bioinformatics-inspired adaptation to Ukkonen’s edit distance calculating algorithm and its applicability towards distributed data mining. *Journal of Software Engineering and Applications*, 1(1):8–12, 2008.
- 18 Dan Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009.
- 19 Petr Kolman and Tomasz Walen. Approximating reversal distance for strings with bounded number of duplicates. *Discret. Appl. Math.*, 155(3):327–336, 2007.
- 20 Petr Kolman and Tomasz Walen. Reversal distance for strings with duplicates: Linear time approximation using hitting set. *Electron. J. Comb.*, 14(1), 2007.
- 21 Christian Komusiewicz, Mateus de Oliveira Oliveira, and Meirav Zehavi. Revisiting the parameterized complexity of maximum-duo preservation string mapping. *Theoretical Computer Science*, 847:27–38, 2020.
- 22 Saeed Mehrabi. Approximating weighted duo-preservation in comparative genomics. In Yixin Cao and Jianer Chen, editors, *Computing and Combinatorics*, pages 396–406, Cham, 2017. Springer International Publishing.
- 23 Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.
- 24 Patsaraporn Somboonsak and Mud-Armeen Munlin. A new edit distance method for finding similarity in Dna sequence. *World Academy of Science, Engineering and Technology, International Journal of Biological, Biomolecular, Agricultural, Food and Biotechnological Engineering*, 5:622–626, 2011.
- 25 Torsten Suel and Nasir Memon. Chapter 13 - algorithms for delta compression and remote file synchronization. In Khalid Sayood, editor, *Lossless Compression Handbook*, Communications, Networking and Multimedia, pages 269–289. Academic Press, San Diego, 2003.
- 26 Krister M. Swenson and Bernard M. E. Moret. Inversion-based genomic signatures. *BMC Bioinform.*, 10(S-1), 2009.

5:12 MPSM with Bounded Alphabet

- 27 MK Vijaymeena and K Kavitha. A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal (MLAIJ)*, 3(1), 2016.
- 28 William E Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. *Proceedings of the Section on Survey Research Methods. American Statistical Association*, pages 354–359, 1990.
- 29 Yao Xu, Yong Chen, Guohui Lin, Tian Liu, Taibo Luo, and Peng Zhang. A $(1.4+\epsilon)$ -approximation algorithm for the 2-max-duo problem. In *28th International Symposium on Algorithms and Computation (ISAAC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.