Further Improvement in Approximating the Maximum Duo-Preservation String Mapping Problem

Brian Brubach(✉)

Department of Computer Science,
University of Maryland–College Park, College Park, MD, USA
bbrubach@cs.umd.edu

Abstract. We present an improved approximation for the Maximum Duo-Preservation String Mapping Problem (MPSM). This problem was introduced in [7] as the complement to the well-studied Minimum Common String Partition problem (MCSP). Prior work also considers the k-MPSM and k-MCSP variants in which each letter occurs at most k times. The authors of [7] showed a k^2-appoximation for $k \geq 3$ and 2-approximation for $k = 2$. A 4-approximation independent of k was shown in [4]. In [4], they also showed that k-MPSM is APX-Hard and achieved approximation ratios of 8/5 for $k = 2$ and 3 for $k = 3$. In this paper, we show an algorithm which achieves a 13/4-approximation for the general MPSM problem using a new combinatorial triplet matching approach. During publication of this paper, [3] presented a local search algorithm yielding 7/2, which falls in between the previous best and this paper. The remainder of the paper has not been altered to reflect this.

Keywords: String algorithms · Polynomial-time approximation · Max Duo-Preservation String Mapping Problem · Min Common String Partition Problem

1 Introduction

String comparison is one of the most fundamental problems in many fields such as bioinformatics and data compression. In computer science, the difference between two strings is often measured by edit distance, the number of edit operations required to transform one string into the other. The most widely known definitions of edit distance include insertion, deletion, and/or substitution operations. However, the more general edit distance with moves problem studied in [10] allows an additional operation wherein an entire block of text is shifted within a string.

B. Brubach—Supported in part by NSF award CCF-1422569.
The author wishes to acknowledge internship mentor Prof. Srinivas Aluru for his support.

M. Frith and C.N.S. Pedersen (Eds.): WABI 2016, LNBI 9838, pp. 52–64, 2016.
DOI: 10.1007/978-3-319-43681-4_5

These shift operations, also known as rearrangements, are especially relevant in biology [8,18]. String comparison can be performed on DNA or protein sequences to estimate how closely related different species are. In data compression, we may want to store many similar strings as a single string along with the edits required to recover all strings. These two applications even overlap naturally in the field of bioinformatics where extremely large datasets of biological sequences are common. For example, the challenge of pan-genome storage is to store many highly similar sequences from the same clade such as a bacterial species.

One way to capture just the "moves" operation is to solve the Minimum Common String Partition problem (MCSP) which seeks to partition two strings into minimum cardinality sets of substrings that are permutations of each other. While the MCSP problem has been heavily studied, the complementary Maximum Duo-Preservation String Mapping Problem (MPSM) is a relatively new and under-explored problem in this area.

1.1 Problem Description

The Maximum Duo-Preservation String Mapping Problem (MPSM) is defined as follows. We are given two strings $A = a_1a_2 \dots a_n$ and $B = b_1b_2 \dots b_n$ of length n such that B is a permutation of A. Let a_i and b_j be the i^{th} and j^{th} characters of their respective strings. A *proper* mapping π from A to B is a one-to-one mapping with $a_i = b_{\pi(i)}$ for all $i = 1, \dots, n$. A *duo* is simply two consecutive characters from the same string. We say that a duo (a_i, a_{i+1}) is *preserved* if a_i is mapped some b_j and a_{i+1} is mapped to b_{j+1}. The objective is to return a proper mapping from the letters of A to the letters of B which preserves the maximum number of duos. Note that the number of duos preserved in each string is identical and by convention we count the number of duos preserved in a single string rather than the sum over both strings. Let OPT_{MPSM} denote the number of duos preserved from a single string in an optimal solution to the MPSM problem. Figure 1 shows an example of an optimal mapping which preserves the maximum possible number of duos.

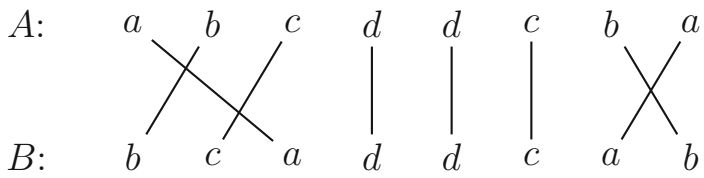

Fig. 1. Illustration of a mapping π from A to B that preserves 3 duos: *bc*, *dd*, and *dc*. A solution to the complementary MCSP problem on the same strings would be partitions $P_A = a, bc, ddc, b, a$ and $P_B = bc, a, ddc, a, b$ with $|P_A| = |P_B| = 5$.

The complementary Minimum Common String Partition problem (MCSP) seeks to find partitions of the strings A and B where a partition P_A of A is defined as a set of substrings whose concatenation is A. The objective is to find minimum cardinality partitions P_A of A and P_B of B such that P_B is a permutation of P_A. Let OPT_{MCSP} denote the cardinality of a partition in an optimal solution to this problem. We can see that

$$OPT_{MCSP} = |P_A| = |P_B| = n - OPT_{MPSM}$$

The variants, k-MPSM and k-MCSP, add the restriction that each letter occurs at most k times in each string. For a given algorithm, let ALG_{MPSM} be number of duos preserved by the algorithm. The approximation ratio for that algorithm is defined as

$$\frac{OPT_{MPSM}}{ALG_{MPSM}}$$

1.2 Related Work

The Maximum Duo-Preservation String Mapping Problem (MPSM) was introduced in [7] along with the related Constrained Maximum Induced Subgraph (CMIS) and Constrained Minimum Induced Subgraph (CNIS) problems. They used a linear programming and randomized rounding approach to approximate the k-CMIS problem which they show is a generalization of k-MPSM. This leads to a k^2-approximation for $k \geq 3$ and a 2-approximation for $k = 2$. This was improved by [4] to a 4-approximation independent of k as well as approximation ratios of 3 for $k = 3$ and 8/5 for $k = 2$. [4] also show that k-MPSM is APX-hard even for $k = 2$, meaning no polynomial-time approximation scheme (PTAS) exists assuming $P \neq NP$. The fixed-parameter tractability was studied in [1] and MPSM was shown to be fixed-parameter tractable when parameterized by the number of preserved duos.

The Minimum Common String Partition problem (MCSP) has been extensively studied from many angles including polynomial-time approximation [7,9,10,14,16,17], fixed-parameter tractability [5,6,11,15], and heuristics [2,12,13]. FPT algorithms have been parameterized by maximum number of times any character occurs, minimum block size, and the size of the optimal minimum partition. Heuristic approaches range from an ant colony optimization algorithm [12] to integer linear programming (ILP) based strategies [2,13] which in some cases solve the problem optimally for strings up to $2,000$ characters in length.

The problem was shown to be NP-hard (thus implying MPSM is also NP-hard) and APX-hard even for 2-MCSP [14]. The current best approximations are an $O(\log n \log^* n)$-approximation due to [10] for general MCSP and an $O(k)$-approximation for k-MCSP due to [17]. Applications to evolutionary distance and genome rearrangement can be found in [8,18].

1.3 Our Contributions

We show a 13/4-approximation ratio for the general MPSM problem using a new combinatorial triplet matching approach. This improves the previous best approximation ratio of 4 for the general problem due to [4].

Theorem 1. *For any two strings A and B such that B is a permutation of A, there is an algorithm which finds a proper mapping from A to B that preserves at least $4/13$ of the duos that the optimal algorithm preserves.*

2 Preliminaries

Let $A = a_1a_2 \dots a_n$ and $B = b_1b_2 \dots b_n$ be the two strings of length n with a_i and b_i being the i^{th} characters of their respective strings. A *duo* $D_i^A = (a_i, a_{i+1})$ corresponds to the pair of consecutive characters a_i and a_{i+1} in the string. We use $D^A = (D_1^A, \dots, D_{n-1}^A)$ and $D^B = (D_1^B, \dots, D_{n-1}^B)$ to denote the sets of *duos* for A and B, respectively. We similarly define a *triplet* $T_i^A = (a_i, a_{i+1}, a_{i+2})$ as a set of three consecutive characters a_i, a_{i+1}, and a_{i+2} in the string and sets of *triplets* $T^A = (T_1^A, \dots, T_{n-2}^A)$ and $T^B = (T_1^B, \dots, T_{n-2}^B)$ for strings A and B, respectively. Observe that the duos D_i^A and D_{i+1}^A correspond to the first two and last two characters, respectively, of the triplet T_i^A. We refer to duos D_i^A and D_{i+1}^A as *subsets* of the triplet T_i^A.

Important note: In the first step of our algorithm, we append a special character '&' to the beginning and end of each string (indices 0 and $n+1$). We define this character to be not equal to any other character including itself (meaning $\& \neq \&$). This ensures that each duo can be a subset of exactly two triplets.

A proper mapping π from A to B is a one-to-one mapping from the letters of A to the letters of B with $a_i = b_{\pi(i)}$ for all $\forall\, i = 1, \dots, n$. Recall that a duo (a_i, a_{i+1}) is preserved if and only if a_i is mapped to some b_j and a_{i+1} is mapped to b_{j+1}. We call a pair of duos (D_i^A, D_j^B) *preservable* if and only if $a_i = b_j$ and $a_{i+1} = b_{j+1}$.

For consistency, we define the concept of conflicting pairs of duos using the terminology of [4] with a small modification to accommodate our particular analysis. Two preservable pairs of duos (D_i^A, D_j^B) and (D_h^A, D_ℓ^B) are said to be *conflicting* if no proper mapping can preserve both of them. These conflicts can be of two types Type 1 and Type 2.

- Type 1: Either $i = h \wedge j \neq \ell$ or $i \neq h \wedge j = \ell$.
- Type 2: Either $i = h + 1 \wedge j \neq \ell + 1$ or $i \neq h + 1 \wedge j = \ell + 1$.

Exception: In our analysis, we also consider two pairs of consecutive preservable duos (D_i^A, D_j^B) and (D_{i+1}^A, D_{j+1}^B) and a third pair of duos (D_h^A, D_ℓ^B) which conflicts with one or both of them, potentially creating conflicts of both Type 1 and Type 2. However, we classify such conflicts simply as Type 1 conflicts.

3 Triplet Matching Approach

In this section, we introduce and analyze the triplet matching algorithm.

3.1 The Triplet Matching Algorithm

We start by finding a weighted matching on triplets that upper bounds the optimal solution, translating that to a fractional matching on duos, and rounding the fractional solution to a mapping S that preserves a number of duos that is at least 4/13 the weight of the triplet matching.

Step 1: Construct a weighted bipartite graph $\mathbf{G_T}$ on the triplets.

We first append the special character '&' to the beginning and end of each string as discussed in the preliminaries. Recall that $\& \neq \&$. This ensures that each duo can be a subset of exactly two triplets.

We then construct a weighted bipartite graph $G_T = (T^A \cup T^B, E)$ with each partition being the set of triplets from a string. We add three types of edges to this graph: *full* edges, *first-half* edges, and *last-half* edges. For a given pair of triplets, (T_i^A, T_j^B), from the different strings, we can add at most one type of edge. A full edge is added if $(a_i = b_j) \wedge (a_{i+1} = b_{j+1}) \wedge (a_{i+2} = b_{j+2})$. A first-half edge is added if $(a_i = b_j) \wedge (a_{i+1} = b_{j+1}) \wedge (a_{i+2} \neq b_{j+2})$. Similarly, a last-half edge is added if $(a_i \neq b_j) \wedge (a_{i+1} = b_{j+1}) \wedge (a_{i+2} = b_{j+2})$. The full edges have weight 1 and the half edges have weight 1/2.

In other words, if the triplets are a perfect match, the weight of the edge is 1. Otherwise, if only the first two or last two characters match, the weight is 1/2. Finally, if the previous conditions are not met, we do not add an edge between these triplets. Figure 2 illustrates this step.

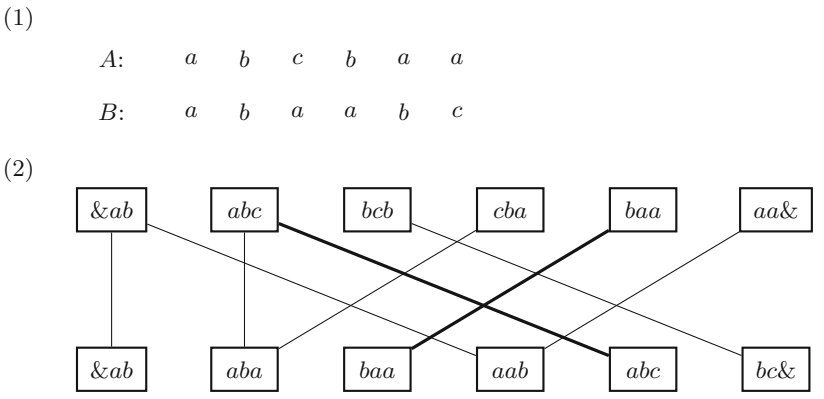

Fig. 2. Step 1 of the algorithm. (1) shows the original strings. (2) shows the bipartite triplet graph G_T.

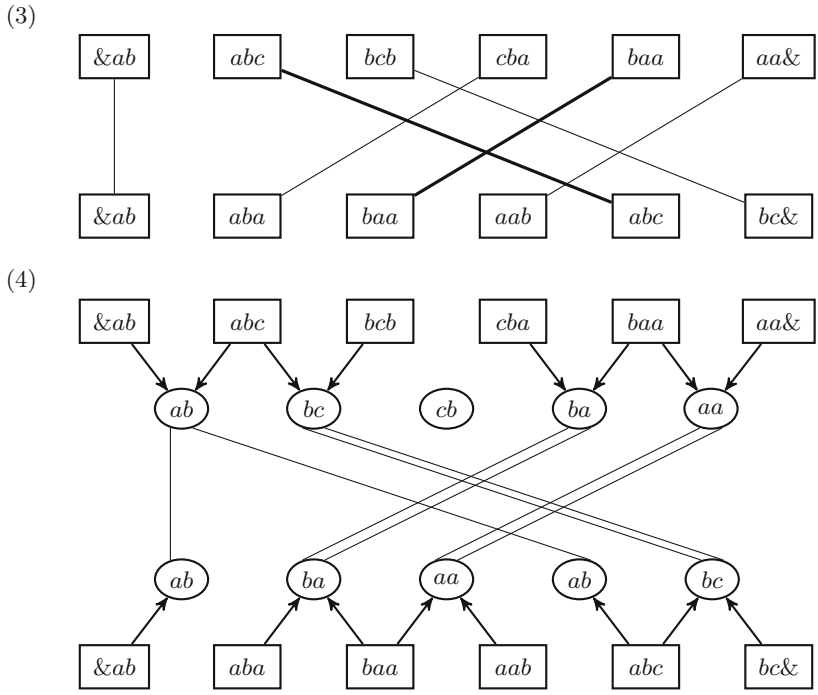

Fig. 3. Steps 2 and 3 of the algorithm. (3) shows the maximum weight matching found in Step 2. (4) shows the construction of the bipartite duo graph G_D in Step 3. Note that the double edges connecting the duos (b, c), (b, a), and (a, a) will each be collapsed into single edges of weight 1.

Step 2: Find a maximum weight matching $\mathbf{M_T}$ on the triplets in $\mathbf{G_T}$.

We find a maximum weight matching M_T in the graph G_T. We will prove later that the weight of this matching is a valid upper bound on the optimum solution to the MPSM problem. Figure 3 illustrates this step.

Step 3: Transfer the matching to a weighted bipartite graph $\mathbf{G_D}$ on the duos.

We now construct a bipartite graph $G_D = (D^A \cup D^B, E)$ on the duos using the edges of the matching M_T found on G_T. For every edge $(T_i^A, T_j^B) \in M_T$, we add one or two edges to G_D. Each edge added to G_D has weight 1/2. Since each duo from the original string is contained in two separate triplets, it can happen that we get two copies of the edge (D_i^A, D_j^B). In this case, we simply merge them into a single edge with weight 1. Edges are added according to the following simple rules:

- If (T_i^A, T_j^B) is a full edge, we add the edges (D_i^A, D_j^B) and (D_{i+1}^A, D_{j+1}^B) to G_D.
- If (T_i^A, T_j^B) is a first-half edge, we add the edge (D_i^A, D_j^B) to G_D.

– If (T_i^A, T_j^B) is a last-half edge, we add the edge (D_{i+1}^A, D_{j+1}^B) to G_D.

Recall that T_i^A and D_i^A refer to the triplet and duo, respectively, starting at letter a_i in the string A and the duos D_i^A and D_{i+1}^A are both subsets of the triplet T_i^A. If the triplet edge (T_i^A, T_j^B) causes duo edges (D_i^A, D_j^B) or (D_{i+1}^A, D_{j+1}^B), we say that the triplets *support* the duo edges. The extra '&' characters are discarded in this step since by definition, they can't be part of any pair of matched duos.

Step 4: Use G_D to find a mapping from string A to string B.

In this step, we select a subset of the edges in G_D to be the duos preserved in our final mapping solution S. This step happens in three phases, each of which may include many iterations. Each iteration of a phase removes edges from G_D corresponding to one or two pairs of duos preserved as well as any conflicting edges. The first two phases each remove all instances of a particular structure from the graph while the third phase tries to preserve as many duos as possible from the remaining graph.

Phase 1. For each edge $(D_i^A, D_j^B) \in G_D$ with weight 1. We remove (D_i^A, D_j^B) from G_D and map a_i and a_{i+1} to b_i and b_{i+1} in S. We also remove any conflicting edges from G_D.

Phase 2. Define a *pair of consecutive parallel edges* to be edges (D_i^A, D_j^B) and (D_{i+1}^A, D_{j+1}^B) in G_D such that the triplet edge (T_i^A, T_j^B) was chosen in M_T. Starting at the beginning of string A, we choose the first pair of consecutive parallel edges (D_i^A, D_j^B) and (D_{i+1}^A, D_{j+1}^B) in G_D. In other words, we find the smallest i such that (D_i^A, D_j^B) and (D_{i+1}^A, D_{j+1}^B) are a pair of consecutive parallel edges. We map a_i, a_{i+1}, and a_{i+2} to b_i, b_{i+1}, and b_{i+2} in S. We then remove the edges (D_i^A, D_j^B) and (D_{i+1}^A, D_{j+1}^B) from G_D as well as any conflicting edges. We continue this process until we reach the end of string A and no pairs of consecutive parallel edges remain in G_D.

Phase 3. Starting at the beginning of string A, we add the duos of the first edge we encounter to S and remove any conflicting edges. We repeat this step until we reach the end of A and no edges remain in G_D.

3.2 Proof of 13/4-approximation

We will first show that the weight of the maximum weight triplet matching M_T found in Step 2 (and by construction the total weight of G_D) is an upper bound on the maximum number of duos preserved. Then, we will show that the number of preserved duos added to S in each iteration of Step 4 is at least 4/13 of the total weight of edges removed from G_D in that iteration. Finally, we will show that at the end of Phase 3 of Step 4, no edges remain in G_D.

Lemma 1. *The weights of the maximum weight triplet matching M_T and the corresponding duo graph G_D are an upper bound on the maximum number of duos preserved.*

Proof. We show that any proper mapping π from A to B which preserves Δ duos implies a matching M_T of weight at least Δ in the corresponding triplet graph G_T.

For each preserved duo (D_i^A, D_j^B) in π, we add the triplet edges (T_{i-1}^A, T_{j-1}^B) and (T_i^A, T_j^B) to M_T if they have not been added already. Note that in the construction of G_T, (D_i^A, D_j^B) was responsible for adding $1/2$ to the weights of both (T_{i-1}^A, T_{j-1}^B) and (T_i^A, T_j^B) for a total contribution of 1. Thus, if we can guarantee that both triplet edges are added to the matching for each preserved duo, that ensures M_T has weight at least Δ.

Assume for the sake of contradiction that we encounter some preserved duo (D_i^A, D_j^B) and at least one of the triplet edges corresponding to (D_i^A, D_j^B) cannot be added. WLOG assume the triplet edge which cannot be added is (T_i^A, T_j^B) and it is blocked by some other edge (T_i^A, T_ℓ^B), $j \neq \ell$. The edge (T_i^A, T_ℓ^B) must have been added by either the preserved duo (D_i^A, D_ℓ^B) or $(D_{i+1}^A, D_{\ell+1}^B)$. However, both of those duos are in conflict with (D_i^A, D_j^B) and therefore could not exist in the mapping π, leading to a contradiction. It follows that both triplet edges are added to the matching for each preserved duo in π. □

Lemma 2. *The number of preserved duos added to S in each iteration of Phase 1 of Step 4 is at least $1/3$ the total weight of edges removed from G_D in that iteration.*

Proof. The worst case structure for this phase is illustrated in Fig. 4.

Suppose some edge (D_i^A, D_j^B) has weight 1 in G_D. Then both triplet edges (T_{i-1}^A, T_{j-1}^B) and (T_i^A, T_j^B) containing (D_i^A, D_j^B) must have been chosen in the matching M_T. Therefore, there can be no conflicts of Type 1.

Note that the edge (D_i^A, D_j^B) can have at most four conflicts of Type 2 arising from the neighboring duos D_{i-1}^A, D_{i+1}^A, D_{j-1}^B, and D_{j+1}^B. Each of these potential conflicts is symmetric. So WLOG, we focus on the conflict with D_{i-1}^A and show that there is at most one edge (D_{i-1}^A, D_ℓ^B), $\ell \neq j-1$, and this edge can only have weight $1/2$.

By construction, the edge (D_{i-1}^A, D_ℓ^B) can only be added by the triplet edges $(T_{i-2}^A, , T_{\ell-1}^B)$ or $(T_{i-1}^A, , T_\ell^B)$. However, the latter triplet edge $(T_{i-1}^A, , T_\ell^B)$ could not exist in M_T since we assume M_T contains the edge (T_{i-1}^A, T_{j-1}^B) and M_T is a matching. Therefore, there is at most one such edge (D_{i-1}^A, D_ℓ^B) with weight $1/2$ which must have come from a triplet edge $(T_{i-2}^A, T_{\ell-1}^B)$ chosen in the matching M_T.

Then the sum of weights of edges removed is at most the weight (D_i^A, D_j^B) plus the weight of four Type 2 conflicting edges with weight $1/2$ each:

$$1 + 4(1/2) = 3$$

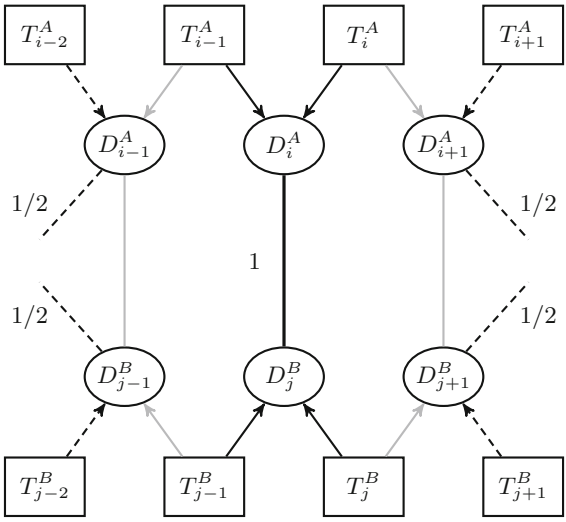

Fig. 4. Illustration of the worst case for Phase 1 of Step 4 in Lemma 2. The solid black lines correspond to the edge (D^A_i, D^B_j) and its supporting triplets. The dashed lines correspond to conflicting edges of Type 2 that must be removed. The gray lines illustrate other edges that may or may not exist, but are not conflicting.

It follows that the ratio of the number of preserved duos added to S to weight of edges removed from G_D is at least $1/3$. □

Note that after Phase 1 of Step 4, all remaining edges in G_D have weight $1/2$ since the edges with weight 1 have been removed.

Lemma 3. *The number of preserved duos added to S in each iteration of Phase 2 of Step 4 is at least $4/13$ of the total weight of edges removed from G_D in that iteration.*

Proof. The worst case structure for this phase is illustrated in Fig. 5.

Suppose we select edges (D^A_i, D^B_j) and (D^A_{i+1}, D^B_{j+1}) in Phase 2. We can upper bound the number of edges removed by identifying all triplets that could support conflicting duo edges and bounding the number of such edges they could have supported. Recall that a triplet supports a duo edge if it belongs to a triplet edge in M_T and thus caused the duo edge to be added to G_D.

First, there are four triplets at distance two from i and j that could each support at most one conflicting edge. These are triplets T^A_{i-2}, T^A_{i+2}, T^B_{j-2}, and T^B_{j+2}. Second, there are four triplets at distance one. Three of these, T^A_{i+1}, T^B_{j-1}, and T^B_{j+1}, can support two conflicting edges. However, the fourth triplet, T^A_{i-1}, can support at most one conflicting edge since we chose the smallest i such that (D^A_i, D^B_j) and (D^A_{i+1}, D^B_{j+1}) are a pair of consecutive parallel edges.

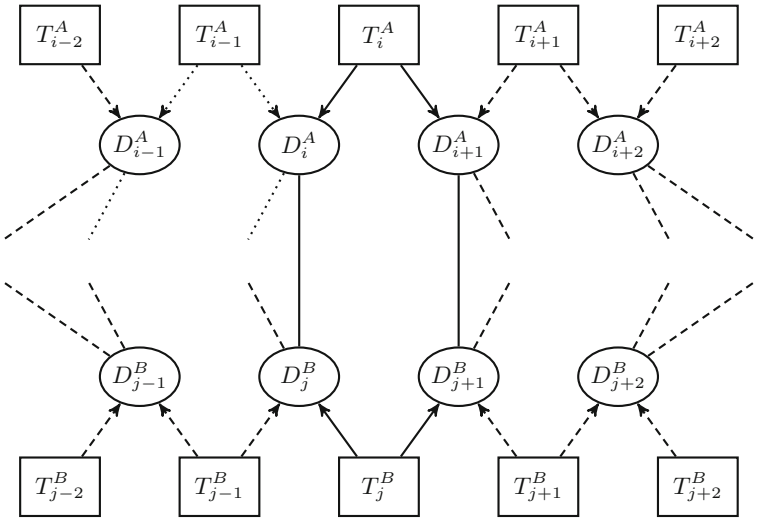

Fig. 5. Illustration of the worst case for Phase 2 of Step 4 in Lemma 3. The solid lines correspond to the edges (D_i^A, D_j^B) and (D_{i+1}^A, D_{j+1}^B) and their supporting triplets. The dashed lines correspond to conflicting edges that must be removed. The pair of parallel dotted lines originating from T_{i-1}^A represent two edges that could not both exist. This is due to the assumption that (D_i^A, D_j^B) and (D_{i+1}^A, D_{j+1}^B) is the first pair of consecutive parallel edges in the string A.

In addition to these conflicting edges, we also remove the two edges (D_i^A, D_j^B) and (D_{i+1}^A, D_{j+1}^B) leading to a total weight removed of

$$4(1/2) + 7(1/2) + 2(1/2) = 6.5$$

It follows that the ratio of the number of preserved duos added to S to weight of edges removed from G_D is at least $2/6.5 = 4/13$. □

Lemma 4. *The number of preserved duos added to S in each iteration of Phase 3 of Step 4 is at least $1/3$ of the total weight of edges removed from G_D in that iteration.*

Proof. Suppose we select the duo edge (D_i^A, D_j^B) in some iteration of Phase 3. We can upper bound the weight of edges deleted from G_D by counting the number of triplets which could have supported duo edges that conflict with (D_i^A, D_j^B). Recall that a triplet supports a duo edge if it belongs to a triplet edge in M_T and thus caused the duo edge to be added to G_D. Because we have removed all pairs of consecutive parallel edges in Phase 2, each triplet can support at most one duo edge remaining in G_D.

There are eight triplets which could potentially support a conflicting duo edge: T^A_{i-2}, T^A_{i-1}, T^A_i, T^A_{i+1}, T^B_{j-2}, T^B_{j-1}, T^B_j, T^B_{j+1}. Notice that we have been selecting edges starting from the beginning of A and moving towards the end. Therefore any edge supported by the triplet T^A_{i-2} would have already been selected or removed prior to the current iteration. Further note that two of those triplets must support the currently selected edge. Therefore, we removed the selected duo edge (D^A_i, D^B_j) and at most five other duo edges. Each of these edges has weight at most $1/2$ since all edges of weight 1 were removed in Phase 1. Then the sum of weights of edges removed is at most

$$1/2 + 5(1/2) = 3$$

It follows that the ratio of the number of preserved duos added to S to weight of edges removed from G_D is at least $1/3$. □

Lemma 5. *At the conclusion of Step 4, no edges remain in G_D.*

Proof. Phase 3 iterates through every remaining edge in G_D, thus removing all of them. □

4 Conclusion and Future Directions

We have shown that a combinatorial triplet matching approach yields an improved approximation to the Maximum Duo Preservation String Mapping problem. Given the fact that triplet matching allows for an improvement over the ratio achieved by the duo matching approach in [4], a natural question is whether a 4-tuple matching could yield even better results. However, a direct extension of the work in this paper to a 4-tuple matching approach is not possible because the 4-tuple matching would not provide an upper bound on the MPSM. The issue with such an approach is that the first and last duos in a 4-tuple have no potential to be conflicting and likely should not be grouped together. On the bright side, we conjecture that the triplet matching approach can be pushed further to achieve a 3-approximation. The clear bottleneck in this paper arises from Phase 2 of Step 4, but we're hopeful this obstacle can be avoided somehow.

Other interesting future directions would be to follow the lead of the work on the MCSP problem. This could include analyzing the performance of faster algorithms such as greedy algorithms or searching for heuristics that solve smaller instances of the problem near optimally. Further, since MPSM currently appears to be "easier" than MCSP, it could be fruitful to explore more applications for this problem in fields such as bioinformatics and data compression.

References

1. Beretta, S., Castelli, M., Dondi, R.: Parameterized tractability of the maximum-duo preservation string mapping problem. CoRR abs/1512.03220 (2015). http://arxiv.org/abs/1512.03220

2. Blum, C., Lozano, J.A., Davidson, P.: Mathematical programming strategies for solving the minimum common string partition problem. Eur. J. Oper. Res. **242**(3), 769–777 (2015). http://www.sciencedirect.com/science/article/pii/S0377221714008716
3. Boria, N., Cabodi, G., Camurati, P., Palena, M., Pasini, P., Quer, S.: A 7/2-approximation algorithm for the maximum duo-preservation string mapping problem. In: Grossi, R., Lewenstein, M. (eds.) CPM 2016. LIPIcs, vol. 54, pp. 11:1–11:8. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl (2016). http://dx.doi.org/10.4230/LIPIcs.CPM.2016.11
4. Boria, N., Kurpisz, A., Leppänen, S., Mastrolilli, M.: Improved approximation for the maximum duo-preservation string mapping problem. In: Brown, D., Morgenstern, B. (eds.) WABI 2014. LNCS, vol. 8701, pp. 14–25. Springer, Heidelberg (2014)
5. Bulteau, L., Fertin, G., Komusiewicz, C., Rusu, I.: A Fixed-parameter algorithm for minimum common string partition with few duplications. In: Darling, A., Stoye, J. (eds.) WABI 2013. LNCS, vol. 8126, pp. 244–258. Springer, Heidelberg (2013)
6. Bulteau, L., Komusiewicz, C.: Minimum common string partition parameterized by partition size is fixed-parameter tractable, Chap. 8, pp. 102–121 (2014). http://epubs.siam.org/doi/abs/10.1137/1.9781611973402.8
7. Chen, W., Chen, Z., Samatova, N.F., Peng, L., Wang, J., Tang, M.: Solving the maximum duo-preservation string mapping problem with linear programming. Theoret. Comput. Sci. **530**, 1–11 (2014). http://www.sciencedirect.com/science/article/pii/S0304397514001108
8. Chen, X., Zheng, J., Fu, Z., Nan, P., Zhong, Y., Lonardi, S., Jiang, T.: Assignment of orthologous genes via genome rearrangement. IEEE/ACM Trans. Comput. Biol. Bioinform. **2**(4), 302–315 (2005)
9. Chrobak, M., Kolman, P., Sgall, J.: The greedy algorithm for the minimum common string partition problem. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) RANDOM 2004 and APPROX 2004. LNCS, vol. 3122, pp. 84–95. Springer, Heidelberg (2004)
10. Cormode, G., Muthukrishnan, S.: The string edit distance matching problem with moves. ACM Trans. Algorithms **3**(1), 2:1–2:19 (2007). http://doi.acm.org/10.1145/1186810.1186812
11. Damaschke, P.: Minimum common string partition parameterized. In: Crandall, K.A., Lagergren, J. (eds.) WABI 2008. LNCS (LNBI), vol. 5251, pp. 87–98. Springer, Heidelberg (2008)
12. Ferdous, S.M., Rahman, M.S.: Solving the minimum common string partition problem with the help of ants. In: Tan, Y., Shi, Y., Mo, H. (eds.) ICSI 2013, Part I. LNCS, vol. 7928, pp. 306–313. Springer, Heidelberg (2013)
13. Ferdous, S.M., Rahman, M.S.: An integer programming formulation of the minimum common string partition problem. PLoS ONE **10**(7), 1–16 (2015)
14. Goldstein, A., Kolman, P., Zheng, J.: Minimum common string partition problem: hardness and approximations. In: Fleischer, R., Trippen, G. (eds.) ISAAC 2004. LNCS, vol. 3341, pp. 484–495. Springer, Heidelberg (2004)
15. Jiang, H., Zhu, B., Zhu, D., Zhu, H.: Minimum common string partition revisited. J. Comb. Optim. **23**(4), 519–527 (2012). http://dx.doi.org/10.1007/s10878-010-9370-2
16. Kolman, P., Waleń, T.: Approximating reversal distance for strings with bounded number of duplicates. Disc. Appl. Math. **155**(3), 327–336 (2007). http://www.sciencedirect.com/science/article/pii/S0166218X0600309X

17. Kolman, P., Waleń, T.: Reversal distance for strings with duplicates: linear time approximation using hitting set. In: Erlebach, T., Kaklamanis, C. (eds.) WAOA 2006. LNCS, vol. 4368, pp. 279–289. Springer, Heidelberg (2007)
18. Swenson, K.M., Marron, M., Earnest-deyoung, J.V., Moret, B.M.E.: Approximating the true evolutionary distance between two genomes. In: Proceedings of 7th SIAM Workshop on Algorithm Engineering and Experiments (ALENEX 2005), p. 121. SIAM Press (2005)