Approximating Weighted Duo-Preservation in Comparative Genomics

Saeed Mehrabi

Cheriton School of Computer Science
University of Waterloo, Waterloo, Canada
smehrabi@uwaterloo.ca

Abstract. Motivated by comparative genomics, Chen et al. [9] introduced the Maximum Duo-preservation String Mapping (MDSM) problem in which we are given two strings s_1 and s_2 from the same alphabet and the goal is to find a mapping π between them so as to maximize the number of duos preserved. A *duo* is any two consecutive characters in a string and it is *preserved* in the mapping if its two consecutive characters in s_1 are mapped to same two consecutive characters in s_2. The MDSM problem is known to be NP-hard and there are approximation algorithms for this problem [3, 5], all of which consider only the "unweighted" version of the problem in the sense that a duo from s_1 is preserved by mapping to any same duo in s_2 regardless of their positions in the respective strings. However, it is well-desired in comparative genomics to find mappings that consider preserving duos that are "closer" to each other under some distance measure [18].

In this paper, we introduce a generalized version of the problem, called the Maximum-Weight Duo-preservation String Mapping (MWDSM) problem, capturing both duos-preservation and duos-distance measures in the sense that mapping a duo from s_1 to each preserved duo in s_2 has a weight, indicating the "closeness" of the two duos. The objective of the MWDSM problem is to find a mapping so as to maximize the total weight of preserved duos. We give a polynomial-time 6-approximation algorithm for this problem.

1 Introduction

Strings comparison is one of the central problems in the field of stringology with many applications such as in Data Compression and Bioinformatics. One of the most common goals of strings comparison is to measure the similarity between them, and one of the many ways in doing so is to compute the *edit distance* between them. The edit distance between two strings is defined as the minimum number of edit operations to transform one string into the other. In biology, during the process of DNA sequencing for instance, computing the edit distance between the DNA molecules of different species can provide insight about the level of "synteny" between them; here, each edit operation is considered as a single mutation.

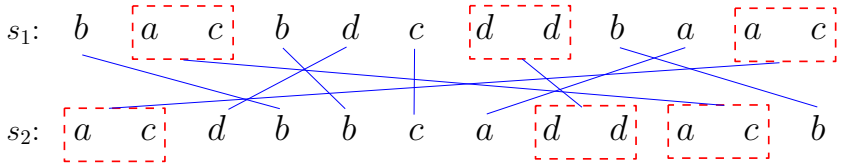


Fig. 1: An instance of the MDSM problem in which the mapping π preserves three duos.

In the simplest form, the only edit operation that is allowed in computing the edit distance is to shift a block of characters; that is, to change the order of the characters in the string. Computing the edit distance under this operation reduces to the Minimum Common String Partition (MCSP) problem, which was introduced by Goldstein et al. [14] (see also [20]) and is defined as follows. For a string s, let $P(s)$ denote a partition of s. Given two strings s_1 and s_2 each of length n, where s_2 is a permutation of s_1, the objective of the MCSP problem is to find a partition $P(s_1)$ of s_1 and $P(s_2)$ of s_2 of minimum cardinality such that $P(s_2)$ is a permutation of $P(s_1)$. The problem is known to be NP-hard and even APX-hard [14].

Recently, Chen et al. [9] introduced a maximization version of the MCSP problem, called the *Maximum Duo-preservation String Mapping* (MDSM) problem. A *duo* in a string s is a pair of consecutive characters in s. For two strings s_1 and s_2, where s_2 is a permutation of s_1 under a mapping π, we say that a *duo is preserved* in the mapping π, if its two consecutive characters are mapped to same two consecutive characters in s_2. Notice that if partitions $P(s_1)$ and $P(s_2)$ are a solution of size r for an instance of the MCSP problem, then this solution can be used to obtain a mapping π from s_1 to s_2 that preserve $n - r$ duos. As such, given two strings s_1 and s_2, the objective of the MDSM problem is to compute a mapping π from s_1 to s_2 that preserves a maximum number of duos. See Figure 1 for an example.

Related Work. Since the MCSP problem is NP-hard [14], there has been many works on designing polynomial-time approximation algorithms for this problem [10–12, 14, 17]. The best approximation results thus far are an $O(\log n \log^* n)$-approximation algorithm for the general version of the problem [12], and an $O(k)$-approximation for the k-MCSP problem [12] (the k-MCSP is a variant of the problem in which each character can appear at most k times in each string). In terms of the parameterized complexity, the problem is known to be fixed-parameter tractable with respect to k and the size of an optimal partition [6, 16], as well as the size of an optimal solution only [7]. For the MDSM problem, we observe that since the MCSP problem is NP-hard [14], the MDSM problem (i.e., its maximization version) is also NP-hard, and in fact even APX-hard [4]. Moreover, the problem is also shown to be fixed-parameter tractable with respect to the number of duos preserved [2]. Boria et al. [4] gave a 4-approximation algorithm for the MDSM problem, which was subsequently improved to algorithms

with approximation factors of 7/2 [3], 3.25 [5] and (recently) $(2+\epsilon)$ for any $\epsilon > 0$ [13].

Motivation and Problem Statement. Observe that in the MDSM problem mapping a duo from s_1 to a preserved duo in s_2 does not consider the position of the two duos in s_1 and s_2. In Figure 1, for instance, the first ac in s_1 is mapped to the second ac in s_2 and the second ac in s_1 is mapped to the first ac in s_2. But, another (perhaps more realistic) mapping would be the one that maps the first ac in s_1 to the first ac in s_2 and the second one in s_1 to the second one in s_2. The latter mapping would become more desirable when solving the problem on strings of extremely long length. In fact, considering the applications of the MDSM problem in comparative genomics, it is much more desirable to find mappings that take into account the position of the preserved features in the two sequences [18, 15]. One reason behind this is the fact that focusing on giving priority to preserving features that are "closer" to each other (distance-wise under some distance measure) provides better information about the "synteny" of the corresponding species [18, 15].

In this paper, we introduce a more general variant of the MDSM problem, called the Maximum-Weight Duo-preservation String Mapping (MWDSM) problem. In this problem, in addition to s_1 and s_2, we are also given a *weight function* defined on pairs of duos that considers the position of the duos in s_1 and s_2, and so better captures the concept of "synteny" in comparative genomics. Now, the objective becomes maximizing the total weight of the preserved duo (instead of maximizing the number of them). Let us define the problem more formally. For a string s, we denote by $D(s)$ the set of all duos in s ordered from left to right. For example, if $s = acbbda$, then $D(s) = \{ac, cb, bb, bd, da\}$.

Definition 1 (The MWDSM Problem). *Let s_1 and s_2 be two strings of length n. Moreover, let $\mathbf{w} : D(s_1) \times D(s_2) \to \mathbb{R}^+$ denote a weight function. Then, the MWDSM problem asks for a mapping π from s_1 to s_2 that preserve a set S of duos such that*

$$\sum_{d \in S} w(d, \pi(d))$$

is maximized over all such sets S, where $\pi(d)$ denotes the duo in s_2 to which $d \in s_1$ is mapped.

We note that the weight function is very flexible in the sense that it can capture any combination of duos-preservation and duos-distance measures. To our knowledge, this is the first formal study of a "weighted version" of the MDSM problem.

Our Result. Notice that the MWDSM problem is NP-hard as its unweighted variant (i.e., the MDSM problem) is known to be NP-hard [14]. We note that the previous approximation algorithms for the MDSM problem do not apply to the MWDSM problem. In particular, both 7/2-approximation algorithm of Boria et al. [3] and $(2+\epsilon)$-approximation algorithm of Dudek et al. [13] are based

on the local search technique, which is known to fail for weighted problems [19, 8]. Moreover, the 3.25-approximation algorithm of Brubach [5] relies on a triplet matching approach, which involves finding a weighted matching (with specialized weights) on a particular graph, but it is not clear if the approach could handle the MWDSM problem with any arbitrary weight function w. Finally, while the linear programming algorithm of Chen et al. [9] might apply to the MWDSM problem, the approximation factor will likely stay the same, which is k^2, where k is the maximum number of times each character appears in s_1 and s_2.

In this paper, we give a polynomial-time 6-approximation algorithm for the MWDSM problem for any arbitrary weight function w. To this end, we construct a vertex-weighted graph corresponding to the MWDSM problem and show that the problem reduces to the *Maximum-Weight Independent Set (MWIS)* problem on this graph. Then, we apply the *local ratio technique* to approximate the MWIS problem on this graph. The local ratio technique was introduced by Bar-Yehuda and Even [1], and is used for designing approximation algorithms for mainly weighted optimization problems (see Section 2 for a formal description of this technique). While the approximation factor of our algorithm is slightly large in compare to that of algorithms for the unweighted version of the problem [3, 5], as we now have weights, our algorithm is much simpler as it benefits from the simplicity of the local ratio technique. To our knowledge, this is the first application of the local ratio technique to problems in stringology.

Organization. The paper is organized as follows. We first give some definitions and preliminary results in Section 2. Then, we present our 6-approximation algorithm in Section 3, and will conclude the paper with a discussion on future work in Section 4.

2 Preliminaries

In this section, we give some definitions and preliminaries. For a graph G, we denote the set of vertices and edges of G by $V(G)$ and $E(G)$, respectively. For a vertex $u \in V(G)$, we denote the set of neighbours of u by $N[u]$; note that $u \in N[u]$.

Let $\mathbf{w} \in \mathbb{R}^n$ be a weight vector, and let F be a set of feasibility constraints on vectors $\mathbf{x} \in \mathbb{R}^n$. A vector $\mathbf{x} \in \mathbb{R}^n$ is a feasible solution to a given problem $(F, \mathbf{p})$ if it satisfies all of the constraints in F. The value of a feasible solution $\mathbf{x}$ is the inner product $\mathbf{w} \cdot \mathbf{x}$. A feasible solution is *optimal* for a maximization (resp., minimization) problem if its value is maximal (resp., minimal) among all feasible solutions. A feasible solution $\mathbf{x}$ is an α-approximation solution, or simply an α-approximation, for a maximization problem if $\mathbf{w} \cdot \mathbf{x} \geq \alpha \cdot \mathbf{w} \cdot \mathbf{x}^*$, where $\mathbf{x}^*$ is an optimal solution. An algorithm is said to have an *approximation factor* of α (or, it is called an *α-approximation algorithm*), if it always computes α-approximation solutions.

Local Ratio. Our approximation algorithm uses the *local ratio technique*. This technique was first developed by Bar-Yehuda and Even [1]. Let us formally state the local ratio theorem.

Theorem 1. *[1] Let F be a set of constraints, and let $\mathbf{w}, \mathbf{w}_1$ and $\mathbf{w}_2$ be weight vectors where $\mathbf{w} = \mathbf{w}_1 + \mathbf{w}_2$. If $\mathbf{x}$ is an α-approximation solution with respect to $(F, \mathbf{w}_1)$ and with respect to $(F, \mathbf{w}_2)$, then $\mathbf{x}$ is an α-approximation solution with respect to $(F, \mathbf{w})$.*

We now describe how the local ratio technique is usually used for solving a problem. First, the solution set is empty. The idea is to find a decomposition of the weight vector $\mathbf{w}$ into $\mathbf{w}_1$ and $\mathbf{w}_2$ such that $\mathbf{w}_1$ is an "easy" weight function in some sense (we will discuss this in more details later). The local ratio algorithm continues recursively on the instance $(F, \mathbf{w}_2)$. We assume inductively that the solution returned recursively for the instance $(F, \mathbf{w}_2)$ is a good approximation and need to prove that it is also a good approximation for $(F, \mathbf{w})$. This requires proving that the solution returned recursively for the instance $(F, \mathbf{w}_2)$ is also a good approximation for the instance $(F, \mathbf{w}_1)$. This step is usually the main part of the proof of the approximation factor.

Graph G_I. Given an instance of the `MWDSM` problem, we first construct a bipartite graph $G_I = (A \cup B, E)$ as follows. The vertices in the left-side set A are the duos in $D(s_1)$ in order from top to bottom and the vertices in the right-side set B are the duos in $D(s_2)$ in order from top to bottom. There exists an edge between two vertices if and only if they represent the same duo. See Figure 2 for an example. Boria et al [4] showed that the `MDSM` problem on s_1 and s_2 reduces to the Maximum Constrained Matching (`MCM`) problem on G_I, which is defined as follows. Let $A = a_1, \ldots, a_n$ and $B = b_1, \ldots, b_n$. Then, we are interested in computing a maximum-cardinality matching M such that if $(a_i, b_j) \in M$, then a_{i+1} can be only matched to b_{j+1} and b_{j+1} can be only matched to a_{i+1}. In the following, we first assign weights to the edges of G_I and will then show that a similar reduction holds between the `MWDSM` problem on s_1 and s_2, and a weighted version of the `MCM` problem on G_I.

To weigh the edges of G_I, we simply assign $w(a_l, b_r)$ as the weight of e, for all $e \in E(G_I)$, where $a_l \in A$ and $b_r \in B$ are the endpoints of e and $w(a_l, b_r)$ is given by Definition 1. Now, we define the Maximum-Weight Constrained Matching (`MWCM`) problem as the problem of computing a maximum-weight matching M in G_I such that if $(a_i, b_j) \in M$, then a_{i+1} can be only matched to b_{j+1} and b_{j+1} can be only matched to a_{i+1}. To see the equivalence between the `MWDSM` problem on s_1 and s_2 and the `MWCM` problem on G_I, let S be a feasible solution to the `MWDSM` problem with total weight $w(S)$ determined by a mapping π. Then, we can obtain a feasible solution S' for the `MWCM` problem on G_I by selecting the edges in G_I that correspond to the preserved duos in S determined by π such that $w(S') = w(S)$. Moreover, it is not too hard to see that any feasible solution for the `MWCM` problem on G_I gives a feasible solution for the `MWDSM` problem with the same weight. This gives us the following lemma.

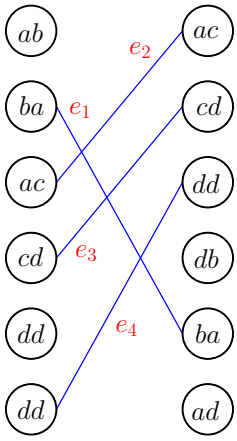


Fig. 2: The graph G_I corresponding to $s_1 = abacddd$ and $s_2 = acddbad$.

Lemma 1. *The* `MWDSM` *problem on* s_1 *and* s_2 *reduces to the* `MWCM` *problem on* G_I.

By Lemma 1, any feasible solution M with total weight $w(M)$ for the `MWCM` problem on G_I gives a mapping π between the strings s_1 and s_2 that preserves a set of duos with total weight $w(M)$. As such, for the rest of this paper, we focus on the `MWCM` problem on G_I and give a polynomial-time 6-approximation algorithm for this problem on G_I, which by Lemma 1, results in an approximation algorithm with the same approximation factor for the `MWDSM` problem on s_1 and s_2.

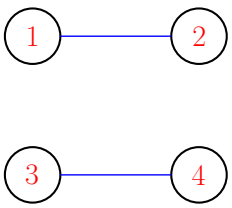
3 Approximation Algorithm

In this section, we give a 6-approximation algorithm for the `MWCM` problem. We were unable to apply the local ratio directly to the `MWCM` problem on G_I due to the constraint involved in the definition of the problem. Instead, we construct a vertex-weighted graph G_C, called the *conflict graph*, and show that the `MWCM` problem on G_I reduces to the Maximum-Weight Independent Set (`MWIS`) problem on G_C. We then apply the local ratio to approximate the `MWIS` problem on G_C, which results in an approximation algorithm for the `MWCM` problem on G_I. Consequently, this gives us an approximation algorithm for the `MWDSM` problem on s_1 and s_2 by Lemma 1.

Graph G_C. We now describe the concept of *conflict*. We say that two edges in $E(G_I)$ are *conflicting* if they both cannot be in a feasible solution for the `MWCM` problem at the same time, either because they share an endpoint or their endpoints are consecutive on one side of the graph, but not on the other side. The following observation is immediate.

Observation 1. *Let $e_1 = (a_i, b_j)$ and $e_2 = (a_k, b_l)$ be two conflicting edges in $E(G_I)$. Then, either $k \in \{i-1, i, i+1\}$ or $l \in \{j-1, j, j+1\}$.*

We define the *conflict graph* G_C as follows. Let $V(G_C)$ be $E(G_I)$; that is, $V(G_C)$ is the set of all edges in G_I. For a vertex $i \in V(G_C)$, we denote the edge in $E(G_I)$ corresponding to i by e_i. Two vertices i and j are adjacent in G_C if and only if e_i and e_j are conflicting in G_I. The conflict graph G_C corresponding to the graph G_I in Figure 2 is shown on the right.



To assign weights to the vertices of G_C, let i be a vertex of G_C. Notice that i corresponds to the edge $e_i = (a_l, b_r)$ in $E(G_I)$, where $a_l \in A$ and $b_r \in B$ are preserved duos. Then, the weight of vertex i is defined as $w(i) := w(a_l, b_r)$ in which recall that $w(a_l, b_r)$ is the weight assigned to these preserved duos by Definition 1. Although not precisely defined, we again note that the weight function is very flexible and it can capture any combination of duos-preservation and duos-distance measures.

Lemma 2. *The* `MWCM` *problem on G_I reduces to the* `MWIS` *problem on G_C.*

Proof. Suppose that S is a feasible solution to the `MWCM` problem on G_I with total weight $w(S)$. For each edge $e_i \in S$, add the vertex $i \in V(G_C)$ to S'. Clearly, S' is an independent set because two vertices i and j in S' being adjacent would imply that e_i and e_j are conflicting in G_I, contradicting the feasibility of S. To see $w(S')$, notice that

$$w(S') = \sum_{i \in S'} w(i) = \sum_{e_i = (a_l, b_r) \in S} w(a_l, b_r) = \sum_{e_i \in S} w(e_i) = w(S).$$

Now, suppose that S' is an independent set in G_C with total weight $w(S')$. For each $u \in S'$, add $e_u \in E(G_I)$ to S. First, S is a feasible solution for the `MWCM` problem on G_I because the vertices of G_C corresponding to any two conflicting edges in S would be adjacent in G_C, contradicting the fact that S' is an independent set. Moreover,

$$w(S) = \sum_{e_i \in S} w(e_i) = \sum_{e_i = (a_l, b_r) \in S} w(a_l, b_r) = \sum_{i \in S'} w(i) = w(S').$$

This completes the proof of the lemma. □

By Lemma 2, any approximation algorithm for the `MWIS` problem on G_C results in an approximation algorithm with the same factor for the `MWCM` problem on G_I. As such, for the rest of this section, we focus on the `MWIS` problem on G_C and show how to apply the local ratio technique to compute a 6-approximation algorithm for this problem on G_C.

Approximating the MWIS Problem on G_C. We first formulate the MWIS problem on G_C as a linear program. We define a variable $x(u)$ for each vertex $u \in V(G_C)$; if $x(u) = 1$, then vertex u belongs to the independent set. The integer program assigns the binary values to the vertices with the constraint that for each clique Q, the sum of the values assigned to all vertices in Q is at most 1.

$$\begin{aligned} &\text{maximize} && \sum_{u \in V(G_C)} w(u) \cdot x(u) && && (1)\\ &\text{subject to} && \sum_{v \in Q} x(v) \leq 1 && \forall \text{ cliques } Q \in G_C, \\ & && x(u) \in \{0,1\} && \forall u \in V(G_C) \end{aligned}$$

Note that the number of constraints in (1) can be exponential in general, as the number of cliques in G_C could be exponential. However, for the MWIS problem on G_C, we can consider only a polynomial number of cliques in G_C. To this end, let $u = (a_i, b_j)$ be a vertex in G_C. By Observation 1, if $v = (a_k, b_l)$ is in conflict with $u = (a_i, b_j)$, then either $k \in \{i-1, i, i+1\}$ or $l \in \{j-1, j, j+1\}$. Let S_u^{i-1} denote the set of all neighbours v of u in G_C such that $v = (a_{i-1}, b_s)$ for some $b_s \in B$ (recall the bipartite graph $G_I = (A \cup B, E)$). Define S_u^i and S_u^{i+1} analogously. Similarly, let S_u^{j-1} be the set of all neighbours v of u such that $v = (a_s, b_{j-1})$ for some $a_s \in A$, and define S_u^j and S_u^{j+1} analogously. Let $M := \{i-1, i, i+1, j-1, j, j+1\}$. Then, by relaxing the integer constraint of the above integer program, we can formulate the MWIS problem on G_C as the following linear program.

$$\begin{aligned} &\text{maximize} && \sum_{u \in V(G_C)} w(u) \cdot x(u) && && (2)\\ &\text{subject to} && \sum_{v \in S_u^r} x(v) \leq 1 && \forall u \in V(G_C), \forall r \in M, \\ & && x(u) \geq 0 && \forall u \in V(G_C) \end{aligned}$$

Notice that the linear program (2) has a polynomial number of constraints. These constraints suffice for the MWIS problem on G_C because, by Observation 1, the vertices $u = (a_i, b_j)$ and v of G_C corresponding to the two conflicting edges e_u and e_v in G_I belong to S_u^r, for some $r \in M$. Moreover, we observe that any independent set in G_C gives a feasible integral solution to the linear program. Therefore, the value of an optimal (not necessarily integer) solution to the linear program is an upper bound on the value of an optimal integral solution.

We are now ready to describe the algorithm. We first compute an optimal solution $\mathbf{x}$ for the above linear program. Then, the rounding algorithm applies a local ratio decomposition of the weight vector $\mathbf{w}$ with respect to $\mathbf{x}$. See Algorithm 1. The key to our rounding algorithm is the following lemma.

Algorithm 1 ApproximateMWIS(G_C)

1: Delete all vertices with non-positive weight. If no vertex remains, then return the empty set;
2: Let $u \in V(G_C)$ be a vertex satisfying

$$\sum_{v \in N[u]} x(v) \leq 6.$$

Then, decompose $\mathbf{w}$ into $\mathbf{w} := \mathbf{w_1} + \mathbf{w_2}$ as follows:

$$w_1(v) := \begin{cases} w(u) & \text{if } v \in N[u], \\ 0 & \text{otherwise.} \end{cases}$$

3: Solve the problem recursively using $\mathbf{w_2}$ as the weight vector. Let S' be the independent set returned;
4: If u is not adjacent with some vertex in S', then return $S := S' \cup \{u\}$; otherwise, return $S := S'$.

Lemma 3. *Let* $\mathbf{x}$ *be a feasible solution to* (2)*. Then, there exists a vertex* $u \in V(G_C)$ *such that*

$$\sum_{v \in N[u]} x(v) \leq 6.$$

Proof. Let $u \in V(G_C)$. Notice that u corresponds to an edge in G_I; that is, $u = (a_i, b_j)$, where a_i and b_j are a pair of preserved duos in the mapping π from s_1 to s_2. Observe that $v \in N[u]$ for some $v = (a_k, b_l) \in V(G_C)$ if and only if (a_k, b_l) conflicts with (i, j) in G_I. Let $M := \{i-1, i, i+1, j-1, j, j+1\}$ and define the set S_u^r as above, for all $r \in M$. Note that the vertices in S_u^r form a clique in G_C, for all $r \in M$, because the set of edges corresponding to the vertices of S_u^r in G_I all share one endpoint (in particular, this endpoint is in A if $r \in \{i-1, i, i+1\}$ or it is in B if $r \in \{j-1, j, j+1\}$). See Figure 3 for an illustration. This means by the first constraint of the linear program (2) that

$$\sum_{v \in S_u^r} x(v) \leq 1,$$

for all $r \in M$. Therefore, we have

$$\sum_{v \in N[u]} x(v) \leq \sum_{r \in M} \sum_{v \in S_u^r} x(v) = |M| = 6.$$

This completes the proof of the lemma. □

We now analyze the algorithm. First, the set S returned by the algorithm is clearly an independent set. The following lemma establishes the approximation factor of the algorithm.

Lemma 4. *Let* $\mathbf{x}$ *be a feasible solution to* (2)*. Then,* $w(S) \geq \frac{1}{6}(\mathbf{w} \cdot \mathbf{x})$.

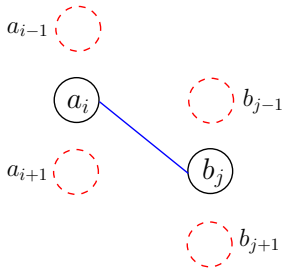


Fig. 3: Graph G_I with edge $u = (a_i, b_j)$. The edge corresponding to any vertex $v \in N[u]$ in G_C is incident to at least one of the six vertices in $\{i-1, i, i+1, j-1, j, j+1$.

Proof. We prove the lemma by induction on the number of recursive calls. In the base case, the set returned by the algorithm satisfies the lemma because no vertices have remained. Moreover, the first step that removes all vertices with non-positive weight cannot decrease the right-hand side of the above inequality.

We now prove the induction step. Suppose that $\mathbf{z}$ and $\mathbf{z}'$ correspond to the indicator vectors for S and S', respectively. By induction, $\mathbf{w_2} \cdot \mathbf{z}' \geq \frac{1}{6}(\mathbf{w_2} \cdot \mathbf{x})$. Since $w_2(u) = 0$, we have $\mathbf{w_2} \cdot \mathbf{z} \geq \frac{1}{6}(\mathbf{w_2} \cdot \mathbf{x})$. From the last step of the algorithm, we know that at least one vertex from $N[u]$ is in S and so we have

$$\mathbf{w_1} \cdot \mathbf{z} = w(u) \sum_{v \in N[u]} z(v) \geq w(u).$$

Moreover, by Lemma 3,

$$\mathbf{w_1} \cdot \mathbf{x} = w(u) \sum_{v \in N[u]} x(v) \leq 6w(u).$$

Hence, $\mathbf{w_1} \cdot \mathbf{x} \leq 6w(u) \leq 6(\mathbf{w_1} \cdot \mathbf{z})$, which gives $\mathbf{w_1} \cdot \mathbf{z} \geq \frac{1}{6}(\mathbf{w_1} \cdot \mathbf{x})$. Therefore, we conclude that $(\mathbf{w_1} + \mathbf{w_2}) \cdot \mathbf{z} \geq \frac{1}{6}(\mathbf{w_1} + \mathbf{w_2}) \cdot \mathbf{x}$ and so $w(S) \geq \frac{1}{6}\mathbf{w} \cdot \mathbf{x}$. This completes the proof of the lemma. □

Since there exists at least one vertex u for which $w_2(u) = 0$ in each recursive step, Algorithm 1 terminates in polynomial time. Therefore, by Lemmas 1, 2 and 4, we have the main result of this paper.

Theorem 2. *There exists a polynomial-time 6-approximation algorithm for the* `MWDSM` *problem on* s_1 *and* s_2.

4 Conclusion

In this paper, we studied a weighted version of the `MDSM` problem [9] that considers the position of the preserved duos in the respective input strings (i.e., the

MWDSM problem). This is a natural variant of the problem, as considering the position of the preserved features in the strings provides solutions with better quality in many applications, such as in comparative genomics in which more weight could indicate more synteny between the corresponding preserved features. We gave a polynomial-time 6-approximation algorithm for the MWDSM problem using the local ratio technique. Although the approximation factor of our algorithm is a bit large in compare to that of algorithms for the unweighted version of the problem, our algorithm is much simpler as it benefits from the simplicity of the local ratio technique. Giving approximation algorithms with better approximation factors for the MWDSM problem remains open for future work.

References

1. Reuven Bar-Yehuda and Shimon Even. A local-ratio theorem for approximating the weighted vertex cover problem. In G. Ausiello and M. Lucertini, editors, *Analysis and Design of Algorithms for Combinatorial Problems*, volume 109, pages 27–45. North-Holland, 1985.
2. Stefano Beretta, Mauro Castelli, and Riccardo Dondi. Parameterized tractability of the maximum-duo preservation string mapping problem. *Theor. Comput. Sci.*, 646:16–25, 2016.
3. 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 *proceedings of the 27th Annual Symposium on Combinatorial Pattern Matching (CPM 2016), Tel Aviv, Israel*, pages 11:1–11:8, 2016.
4. Nicolas Boria, Adam Kurpisz, Samuli Leppänen, and Monaldo Mastrolilli. Improved approximation for the maximum duo-preservation string mapping problem. In *proceedings of the 14th International Workshop on Algorithms in Bioinformatics (WABI 2014), Wroclaw, Poland*, pages 14–25, 2014.
5. Brian Brubach. Further improvement in approximating the maximum duo-preservation string mapping problem. In *proceedings of the 16th International Workshop on Algorithms in Bioinformatics (WABI 2016), Aarhus, Denmark*, pages 52–64, 2016.
6. Laurent Bulteau, Guillaume Fertin, Christian Komusiewicz, and Irena Rusu. A fixed-parameter algorithm for minimum common string partition with few duplications. In *proceedings of the 13th International Workshop on Algorithms in Bioinformatics (WABI 2013), Sophia Antipolis, France*, 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), Portland, Oregon, USA*, pages 102–121, 2014.
8. Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012.
9. 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. *Theor. Comput. Sci.*, 530:1–11, 2014.
10. 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. Biology Bioinform.*, 2(4):302–315, 2005.

11. Marek Chrobak, Petr Kolman, and Jirí Sgall. The greedy algorithm for the minimum common string partition problem. In *proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2004), Cambridge, MA, USA*, pages 84–95, 2004.
12. Graham Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. *ACM Trans. Algorithms*, 3(1):2:1–2:19, 2007.
13. Bartlomiej Dudek, Pawel Gawrychowski, and Piotr Ostropolski-Nalewaja. A family of approximation algorithms for the maximum duo-preservation string mapping problem. *CoRR*, abs/1702.02405, 2017.
14. Avraham Goldstein, Petr Kolman, and Jie Zheng. Minimum common string partition problem: Hardness and approximations. *Electr. J. Comb.*, 12, 2005.
15. Ross C. Hardison. Comparative genomics. *PLoS Biol.*, 1(2):e58, 2003.
16. Haitao Jiang, Binhai Zhu, Daming Zhu, and Hong Zhu. Minimum common string partition revisited. *J. Comb. Optim.*, 23(4):519–527, 2012.
17. Petr Kolman and Tomasz Walen. Reversal distance for strings with duplicates: Linear time approximation using hitting set. *Electr. J. Comb.*, 14(1), 2007.
18. Arcady R. Mushegian. *Foundations of Comparative Genomics*. Academic Press (AP), 2007.
19. Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.
20. Krister M. Swenson, Mark Marron, Joel V. Earnest-DeYoung, and Bernard M. E. Moret. Approximating the true evolutionary distance between two genomes. *ACM Journal of Experimental Algorithmics*, 12:3.5:1–3.5:17, 2008.