An improved approximation algorithm for the complementary maximal strip recovery problem

Zhong Li* Randy Goebel*† Lusheng Wang‡ Guohui Lin*§

January 29, 2011

Abstract

Given two genomic maps G_1 and G_2 each represented as a sequence of n gene markers, the *maximal strip recovery* (MSR) problem is to retain the maximum number of markers in both G_1 and G_2 such that the resultant subsequences, denoted as G_1^* and G_2^*, can be partitioned into the same set of maximal strips, which are common substrings of length greater than or equal to two. The *complementary* maximal strip recovery (CMSR) problem has the complementary goal to delete the minimum number of markers. Both MSR and CMSR have been shown NP-hard and APX-complete, and they admit a 4-approximation and a 3-approximation respectively. In this paper, we present an improved $\frac{7}{3}$-approximation algorithm for the CMSR problem, with its worst-case performance analysis done through a *sequential amortization*.

Keywords: Maximal strip recovery, approximation algorithm, sequential amortized analysis

1 Introduction

In comparative genomics, one of the first steps is to decompose two given genomes into synthetic blocks — segments of chromosomes that are deemed homologous in the two input genomes. Many decomposition methods have been proposed, but they are vulnerable to ambiguities and errors, which are isolated points that do not co-exist with other points in genomic maps [4, 9]. The *maximal strip recovery* (MSR) problem was formulated for eliminating these noise and ambiguities. In the more precise formulation, we are given two genomic maps G_1 and G_2 each represented as a sequence of n distinct gene markers (which form the alphabet Σ), and we want to retain the maximum number of markers in both G_1 and G_2 such that the resultant subsequences, denoted as G_1^* and G_2^*, can be partitioned into the same set of maximal strips, which are common substrings of length greater than or equal to two. Each retained marker thus belongs to exactly one of these substrings, which can appear in the reversed and negated form and are taken as nontrivial chromosomal segments. The deleted markers are regarded as noise or errors.

*Department of Computing Science, University of Alberta. Edmonton, Alberta T6G 2E8, Canada. Email: zhong4@ualberta.ca

†Email: rgoebel@ualberta.ca

‡Department of Computer Science, City University of Hong Kong. Kowloon, Hong Kong, China. Email: cswangl@cityu.edu.hk

§Correspondence author. Email: guohui@ualberta.ca

The MSR problem, and its several close variants, have been shown NP-hard [8, 2, 3]. More recently, it is shown to be APX-complete [2, 6], admitting a 4-approximation algorithm [3]. This 4-approximation algorithm is a modification of an earlier heuristics for computing a maximum clique (and its complement, a maximum independent set) [4, 9], to convert the MSR problem to computing the maximum independent set in t-interval graphs, which admits a $2t$-approximation [1, 3]. In this paper, we investigate the complementary optimization goal to minimize the number of deleted markers — the *complementary MSR* problem, or CMSR for short. CMSR is certainly NP-hard, and was proven to be APX-complete recently [7], admitting a 3-approximation algorithm [5]. Our main result is an improved $\frac{7}{3}$-approximation algorithm for CMSR. As we will show later, the key design technique is a local greedy scheme to retain the most possible isolates while deleting some isolate(s); and the performance ratio is proven using a novel technique called *delayed sequential amortized analysis*. Some preliminary ideas of the design and analysis have appeared in [5].

2 Preliminaries

In the sequel, we use a lower case letter to denote a gene marker. A negation sign together with the succeeding gene indicate that the gene is in its reversed and negated form. We reserve the bullet symbol "$\bullet$" for connection use, that is, $a \bullet b$ means gene b comes directly after gene a (in a specific sequence). When a common substring (also called *strip*, or *synthetic block*) of the two current target sequences (G_1 and G_2, or their remainders after deleting some letters) is specified, it is of length greater than or equal to two, unless otherwise explicitly stated that it is a single letter; The substring will (often) be labeled using a capital letter. We abuse this capital letter a bit to also denote the set of gene markers in the substring, when there is no ambiguity. We present several important structural properties of the CMSR problem in this section, which are used in the design of the approximation algorithm and its performance analysis in the next section.

We first look at a warm-up instance. In this instance, $G_1 = \langle a, b, c, d, e, f, g, h, i, j, k, \ell \rangle$ and $G_2 = \langle -i, -d, -g, -f, h, a, c, b, -\ell, -k, -j, -e \rangle$ (commas are used to separate the gene markers for easier reading). By deleting markers c, d, e, and h from both G_1 and G_2, the remainder sequences are $G_1^* = \langle a, b, f, g, i, j, k, \ell \rangle$ and $G_2^* = \langle -i, -g, -f, a, b, -\ell, -k, -j \rangle$. These two remainder sequences can be decomposed into three maximal common substrings $S_1 = a \bullet b$, $S_2 = f \bullet g \bullet i$ (appearing in the reversed and negated form in G_2^*), and $S_3 = j \bullet k \bullet \ell$ (appearing in the reversed and negated form in G_2^*). For this small instance, one can prove that the optimal solution to the MSR problem has size 8, and (consequently) the optimal solution to the CMSR problem has size 4.

In the rest of the paper, we use OPT to denote an optimal solution to the instance of the CMSR problem. That is, OPT is a minimum-size subset of letters that, deleting them from G_1 and G_2 gives the remainder sequences, denoted G_1^* and G_2^*, respectively, which can be partitioned into maximal common substrings.

Given any CMSR instance, in at most quadratic time, we can determine all maximal common substrings of G_1 and G_2 (of length at least two) and the isolated letters that do not belong to any of the common substrings. We use *unit* to refer to a maximal common substring or an isolated letter. A unit and its reversed and negated form are considered identical. The above determined units form a common partition of G_1 and G_2, *i.e.*, every letter occurs in exactly one of these units. For ease of presentation, these maximal common substrings are called *type*-0 substrings; the isolated

letters are called *isolates*. Note that every letter appears exactly once in each of the two sequences G_1 and G_2; for distinction purpose we use an *isolate copy* to refer to exactly one copy of the isolate. In our algorithm Approx-CMSR to be presented in the next section, all the type-0 substrings are kept in the final sequences and our goal is to eliminate the isolates, by either deleting them from the input sequences, or to "*merge*" them into substrings while deleting some other isolates from the input sequences. Here "merging" refers to either appending an isolate to some existing substring, or forming two isolates into a novel common substring, which is called a *type*-1 substring.

Lemma 1 [5] *For any CMSR instance, there exists an optimal solution OPT such that*

1) for each type-0 substring S, either $S \subset OPT$ or $S \cap OPT = \emptyset$;

2) if $|S| \geq 4$, then $S \cap OPT = \emptyset$.

The above Lemma 1 tells that in the optimal solution, for every type-0 substring, either all its letters are deleted or none of them is deleted. We partition OPT into a subset O_3 of length-3 type-0 substrings, a subset O_2 of length-2 type-0 substrings, and a subset O_1 of isolates: $OPT = O_3 \cup O_2 \cup O_1$. These substrings and isolates are referred to as units of OPT in the sequel.

2.1 Favorable operations

Consider an isolate x in the given sequences G_1 and G_2. For ease of presentation we append an *imaginary* type-0 substring H to the head of G_1 and to the tail of G_2, and another *imaginary* type-0 substring T to the tail of G_1 and to the head of G_2. It follows that x has exactly four neighboring letters in G_1 and G_2 — duplicates are counted separately. Assume without loss of generality that the two neighboring letters in G_1 are a and b, and the two neighboring letters in G_2 are c and d. In the most extreme case, $\{a, b, c, d\}$ reduces to a set of two isolates (see the first row in Table 1) and deleting x enables the merging a and b into a novel length-2 substring. (*Remark*: There is another possible configuration where $a \bullet x \bullet b$ appears in G_1 and $a \bullet -x \bullet b$ appears in G_2, which can be identically discussed. For the same reason, in the following operations, reversed and negated gene form is skipped.) Such a scenario is favorable in the sense that a *gain* of one letter is achieved — deleting one while keeping two. Our algorithm Approx-CMSR will execute the deleting and the subsequent merging, referred to as an operation 1, which is given the top priority.

In another interesting case, $a \bullet b$ appears in G_2 and $c \bullet d$ appears in G_1. When these four letters are distinct isolates (see the second row in Table 1), then deleting x enables the merging of a and b and the merging of c and d into two novel length-2 substrings. Such a scenario is also favorable for a gain of three letters. Our algorithm Approx-CMSR will execute the deleting and the subsequent merging, referred to as an operation 2, which is also given the top priority.

If exactly one of these four letters, say d, resides in a substring S and the other three are distinct isolates (see the third row in Table 1), then deleting x enables the merging of a and b and the appending of c to S. This scenario is also favorable in the sense that a gain of two letters is achieved. Our algorithm Approx-CMSR will execute the deleting and the subsequent merging and appending, referred to as an operation 3, which is given the top priority too. If exactly two of these four letters reside in substrings (see the 4th row in Table 1) such that deleting x enables two separate appending of the neighboring isolates to the two substrings respectively, it is also favorable

for a gain of one letter. Approx-CMSR will execute the deleting and the subsequent appending, referred to as an operation 4, which is given the top priority too.

Priority	Operation	Local configuration	Comments
1	1	$\ldots a_j \bullet x_j \bullet b_j \ldots$ $\ldots - b_j \bullet x_j \bullet -a_j \ldots$	$U_j = \{x_j\}$; $V_j = \{a_j, b_j\}$
	2	$\ldots a_j \bullet x_j \bullet b_j \ldots c_j \bullet d_j \ldots$ $\ldots a_j \bullet b_j \ldots c_j \bullet x_j \bullet d_j \ldots$	a_j, b_j, c_j, d_j are distinct; $U_j = \{x_j\}$; $V_j = \{a_j, b_j, c_j, d_j\}$
	3	$\ldots a_j \bullet x_j \bullet b_j \ldots c_j \bullet S \ldots$ $\ldots a_j \bullet b_j \ldots c_j \bullet x_j \bullet S \ldots$	a_j, b_j, c_j are distinct; $U_j = \{x_j\}$; $V_j = \{a_j, b_j, c_j\}$
	4	$\ldots a_j \bullet x_j \bullet S_1 \ldots c_j \bullet S_2 \ldots$ $\ldots a_j \bullet S_1 \ldots c_j \bullet x_j \bullet S_2 \ldots$	$U_j = \{x_j\}$; $V_j = \{a_j, c_j\}$
	5	$\ldots a_j \bullet x_j \bullet S \ldots$ $\ldots a_j \bullet S \ldots$	$U_j = \{x_j\}$; $V_j = \{a_j\}$
	6	$\ldots a_j \bullet x_j \bullet b_j \bullet y_j \bullet c_j \ldots d_j \bullet e_j \ldots$ $\ldots a_j \bullet b_j \bullet c_j \ldots d_j \bullet x_j \bullet y_j \bullet e_j \ldots$	a_j, b_j, c_j, d_j, e_j are distinct; $U_j = \{x_j, y_j\}$; $V_j = \{a_j, b_j, c_j, d_j, e_j\}$
2	7	$\ldots a_j \bullet x_j \bullet b_j \ldots$ $\ldots a_j \bullet b_j \ldots$	$U_j = \{x_j\}$; $V_j = \{a_j, b_j\}$
3	8	$\ldots a_j \bullet x_j \bullet y_j \bullet b_j \ldots$ $\ldots a_j \bullet b_j \ldots$	$\{x_j, y_j\}$ is an eliminating pair; $U_j = \{x_j, y_j\}$; $V_j = \{a_j, b_j\}$
	9	$\ldots a_j \bullet x_j \bullet b_j \ldots$ $\ldots a_j \bullet y_j \bullet b_j \ldots$	$U_j = \{x_j, y_j\}$; $V_j = \{a_j, b_j\}$

Table 1: Nine different isolate elimination operations with four levels of priorities, where a smaller number indicates a higher priority. In these operations, lower case letters are isolates, while the capital ones are existing substrings.

There are two other top priority operations. In an operation 5, deleting x enables one appending of a neighboring isolate to a substring, for zero gain of letters (see the 5th row in Table 1); In an operation 6, the neighborhood of isolate x overlaps with the neighborhood of another isolate y (see the 6th row in Table 1), such that deleting x and y simultaneously enables the formation of a novel length-2 substring and a novel length-3 substring, using the involved five distinct neighboring isolates in the union of the two neighborhoods. An operation 6 has a gain of three letters.

Notice that wherever an operation 6 can be applied, deleting one and only one of x, y, and b enables a merging of two its neighboring isolates into a novel length-2 substring. For example, from the 6th row in Table 1, deleting b enables the merging of x and y. It is however very important

to recognize that in this scenario both x and y should be deleted. The operation 7, in which one isolate is deleted and its two neighboring isolates are merged into a novel length-2 substring (see the 7th row in Table 1), has a lower priority than all the above six operations. This might look odd since an operation 7 has a gain of one letter, seemingly better than an operation 5; but giving it a lower priority is crucial in the proof of performance guarantee in Lemma 15, where we cannot afford to execute an operation 7 to delete isolate b and form substring $x \bullet y$, which prevents the formation of substrings $a \bullet c$ and $d \bullet e$.

The last two operations have the lowest priority, in both of which two isolates x and y are deleted and (only) one novel length-2 substring is formed using their two neighboring isolates a and b. In an operation 8, x and y reside in between a and b in one sequence, while a and b are adjacent in the other sequence (see the 8th row in Table 1); In an operation 9, x and y reside in between a and b separately one in each of the two sequences (see the 9th row in Table 1). These two operations must have a lower priority than an operation 7, a condition used in the proofs of their performance guarantees in Lemmas 16 and 17.

3 A $\frac{7}{3}$-approximation algorithm

We assume at hand an optimal solution OPT stated in Lemma 1, and it is partitioned into O_3, O_2, and O_1.

3.1 The Approx-CMSR algorithm

In the first step of our approximation algorithm, denoted as Approx-CMSR, it retains all type-0 substrings. That is, Approx-CMSR will only delete isolates from the input sequences (in the second and the third steps).

In the second step, Approx-CMSR iteratively removes one or two isolates; the candidate isolates have to be in one of the nine cases listed in Table 1, and the one with the top priority is chosen (tie breaks arbitrarily) for removal at the time of consideration. In each of the nine cases, the isolate removal can give rise to a novel length-2 or length-3 common substring to the remainder sequences (all except operations 4 and 5), and/or allow an isolate to be appended to an existing common substring in the remainder sequences (operations 3, 4, and 5). If a novel common substring is formed, it is referred to as a type-1 substring; if an existing substring is extended, it retains its type for ease of presentation. The involved isolates, in either scenario, lose their isolate identity and are retained by Approx-CMSR.

Let $\mathcal{U} = \{U_1, U_2, \ldots, U_m\}$, where U_j denotes the set of isolates deleted by Approx-CMSR in the j-th iteration of the second step. Associated with each U_j, for $j = 1, 2, \ldots, m$, let V_j denote the set of isolates that are retained by Approx-CMSR in the j-th iteration. In Table 1, the U_j and V_j for each of the nine operations are specified. Let R denote the set of remaining isolates at the time the algorithm finds no isolates to delete. In the third (the last) step of the algorithm, Approx-CMSR deletes all letters of R from the two sequences. A high-level description of the algorithm Approx-CMSR is depicted in Figure 1.

Algorithm Approx-CMSR:
Input: two sequences (permutations) G_1 and G_2 on the same set of letters. Output: two subsequences of G_1 and G_2 respectively, that can be partitioned into maximal common substrings of length at least 2. 1. Determines all type-0 substrings of G_1 and G_2, and retains them; 2. While (there are feasible operations in the current sequences G_1 and G_2), do 2.1. finds an operation of the currently highest priority; 2.2. removes the letters of U_j from G_1 and G_2; 2.3. retains the letters of V_j in G_1 and G_2 by forming appropriate substrings; 3. Deletes all the remaining isolates from G_1 and G_2.

Figure 1: A high-level description of the algorithm Approx-CMSR.

3.2 Performance analysis

Let $U = \cup_{j=1}^{m} U_j$ and $V = \cup_{j=1}^{m} V_j$. The following two lemmas state some preliminary observations on algorithm Approx-CMSR.

Lemma 2 *The set of all isolates is the union of the disjoint sets $U_1, U_2, \ldots, U_m$, $V_1, V_2, \ldots, V_m$, and R, that is, $U \cup V \cup R$; Algorithm* Approx-CMSR *deletes all isolates of $U \cup R$, but no others.*

Lemma 3 (Once adjacent, always adjacent) *In the j-th iteration of algorithm* Approx-CMSR*, for $j = 1, 2, \ldots, m$, if two letters a_j and b_j (at least one of them is an isolate of V_j) are made adjacent into a common substring, then a_j and b_j are maintained adjacent toward the termination of the algorithm. Moreover, in the two original input sequences G_1 and G_2, all the letters in between a_j and b_j belong to $\cup_{i=1}^{j} U_i$.*

Lemma 4 *In the j-th iteration of algorithm Approx-CMSR, for $j = 1, 2, \ldots, m$, assume two letters a_j and b_j are made adjacent into a common substring. If both a_j and b_j are kept in G_1^* and G_2^* but they are not adjacent to each other, then they do not reside in the same substring of G_1^* and G_2^*.*

Proof. Suppose to the contrary that a_j and b_j reside in the same substring of G_1^* and G_2^* but they are not adjacent to each other. Let $z_1, z_2, \ldots, z_k$ be the letters in between a_j and b_j in G_1^* and G_2^*, for some $k \geq 1$. Note that $\{z_1, z_2, \ldots, z_k\} \subset \cup_{i=1}^{j} U_i$, from Lemma 3.

Assume first $z_1 \in U_i$ for some $i < j$. In the i-th iteration of algorithm Approx-CMSR, there is an adjacency $a_i \bullet b_i$ been made after deleting some letters of U_i, including z_1. It follows that if z_1 were in between a_j and b_j in both original sequences G_1 and G_2, then a_i and b_i would also be in between a_j and b_j in at least one of G_1 and G_2, preventing the j-th iteration to happen. Therefore, $z_1 \in U_j$. This implies that in the j-th iteration of algorithm Approx-CMSR, an operation 1 is executed. Nevertheless, the reversed and negated form of either z_1 or a_j in exactly one of G_1 and G_2 contradicts the substring existence. □

We use W to denote the set of such isolates $a_j \in V_j - O_1$, which is adjacent to b_j in the Approx-CMSR output but does not reside with b_j in the same substring of G_1^* and G_2^*. We have

the following lemma.

Lemma 5 *For any $j = 1, 2, \ldots, m$, if $O_1 \cap (U_j \cup V_j) = \emptyset$, then $V_j \subset W$.*

PROOF. For each $a_j \in V_j$, if its corresponding b_j, to which a_j is made adjacent in the j-th iteration of algorithm Approx-CMSR, is not kept in G_1^* and G_2^*, then $a_j \in W$. If b_j is also kept in G_1^* and G_2^*, then a_j and b_j do not reside in the same substring due to the fact that there is a letter of U_j in between a_j and b_j in exactly one of G_1^* and G_2^*, when an operation 2–9 is executed, and thus $a_j \in W$ by Lemma 4; when an operation 1 is executed in the j-th iteration of algorithm Approx-CMSR, a_j and b_j do not reside in the same substring due to the reversed and negated form of either a_j or x_j. □

In the sequel, we estimate the size of $U \cup R$ in terms of the size of OPT. We do this by attributing all isolates of $U \cup R$ and some isolates of W to the letters of OPT, through a sequential amortized analysis. Note that isolates of $W \subset V$ are not deleted by algorithm Approx-CMSR. When an isolate x of $U \cup R$ and an isolate w of W are identified as a pair, we say *they cancel out each other* in the sense that, to whichever letter $o \in OPT$ w is attributed, w is replaced by x at the end of the attribution process. That is, these attributed isolates of W relay the isolates of $U \cup R$ during the attribution process. We create three *bins* to classify the letters of OPT. A letter of OPT is classified into bin B_2 if it is attributed with at most two isolates of $U \cup R \cup W$; a letter of OPT is classified into bin B_3^1 if it is attributed with at most three isolates of $U \cup R \cup W$, and it is associated with a unique isolate copy of $O_1 \cap R$; and a letter of OPT is classified into bin B_3^2 if it is attributed with at most three isolates of $U \cup R \cup W$, and it is associated with two unique isolate copies of $O_1 \cap R$. Note that an isolate of $O_1 \cap R$ has two copies, one in each of G_1 and G_2, and thus can be associated with two letters of $B_3^1 \cup B_3^2$.

Consider the inverse process of deleting units of OPT from G_1 and G_2 to obtain the optimal subsequences G_1^* and G_2^*. In this inverse process, we add the units of OPT back to G_1^* and G_2^* using their original positions in G_1 and G_2 to re-construct G_1 and G_2. At the beginning of this process, there are no isolated letters in G_1^* and G_2^*; each isolate of $U \cup V \cup R$ thus either is a unit of O_1, or is in some substring of G_1^* and G_2^* but then *singled* out by inserting a unit of OPT back into G_1^* and G_2^*, which breaks the substring (or one of its fragments if already broken) into fragments, one of which is the single isolate. In either case, when the isolate is known to be in $U \cup R \cup W$, it is *generated* by the inserting unit of OPT and is thus *attributed* to the unit. Since there could be multiple units of OPT that are able to single out this particular isolate, we will set up the conditions for proper attribution.

At any time of the process, inserting one unit of OPT back to the current G_1 and G_2 can generate at most four single-letter fragments, since in the worst case two current length-2 substrings, one in each of current sequences G_1 and G_2, can be broken into four such fragments. We firstly insert units of O_3 and O_2, one by one; all the isolates of $U \cup R \cup W$ that are singled out by inserting a unit of $O_3 \cup O_2$ are attributed to the unit. Lemma 6 summarizes this fact that every letter in a substring of $O_3 \cup O_2$ is classified into B_2. The resultant sequences are denoted as G_1^0 and G_2^0.

Lemma 6 *Every letter in a substring of $O_3 \cup O_2$ is in B_2.*

PROOF. Inserting one substring of $O_3 \cup O_2$ back into the current sequences G_1 and G_2 generates at most four isolated letters, and every substring contains at least two letters. Therefore, every letter can be attributed with at most two isolated letters of $U \cup R \cup W$. □

Consider an isolate $x \in U \cup R$ that resides in a common substring of G_1^0 and G_2^0. Let $x \bullet y$ be the adjacency (or one of the two adjacencies). Since inserting all units of O_1 has to single x out, we conclude that there are units of O_1 residing in between x and y in the original sequences G_1 and G_2. Denote this subset of units of O_1 as X, inserting each of which back into G_1^0 and G_2^0 will break the adjacency between x and y. If X intersects with $U \cup V$, let j^* denote the maximum j for which $X \cap (U_j \cup V_j) \neq \emptyset$, and we say that the adjacency $x \bullet y$ is broken by the isolates of $O_1 \cap (U_{j^*} \cup V_{j^*})$. If X does not intersect with $U \cup V$, *i.e.*, $X \subset R$, then the adjacency $x \bullet y$ is broken by the isolates of $O_1 \cap R$.

We next insert isolates of $O_1 \cap (U_j \cup V_j)$ back into G_1^0 and G_2^0, for $j = 1, 2, \ldots, m$, sequentially, and show that they all belong to bins $B_2 \cup B_3^1 \cup B_3^2$. Right after inserting isolates of $O_1 \cap (U_j \cup V_j)$, the achieved sequences are denoted as G_1^j and G_2^j. We emphasize that this sequential order is very important, as we need it in the proofs of Lemmas 9–17, each of which counts the number of isolates of $U \cup R \cup W$ that are attributed to the isolates of $O_1 \cap (U_j \cup V_j)$, when one of the nine different operations is executed in the j-th iteration of Approx-CMSR.

Lemma 7 *For any $j = 1, 2, \ldots, m$,*

1) *a letter of U_j and a letter not of U_j cannot reside in the same substring of G_1^j and G_2^j;*
2) *except that Approx-CMSR executes an operation 8 in the j-th iteration, letters of U_j are isolates in G_1^j and G_2^j.*

PROOF. Notice that all type-0 substrings, and all letters of $\cup_{i=1}^{j} (U_i \cup V_i)$ are in G_1^j and G_2^j, guaranteed by the inserting process. We prove this lemma by (finite) induction.

In the base case where $j = 1$, assume Approx-CMSR made an adjacency $a_1 \bullet b_1$. From Lemma 3 we know that the letters in between a_1 and b_1 in the original sequences G_1 and G_2 are all in U_1. It follows that none of these letters resides in a substring of G_1^1 and G_2^1 that contains a letter $z \notin U_1$, since otherwise z must be one of a_1 and b_1, contradicting to all nine defined operations. Note that only an operation 8 can result in $U_1 = \{x_1, y_1\}$ such that both x_1 and y_1 are in between a_1 and b_1 in one of G_1^1 and G_2^1 and not separated in the other sequence. That is, letters of U_1 are isolates in G_1^1 and G_2^1 if they are not the result of an operation 8.

Assume the lemma holds for all $i = 1, 2, \ldots, j-1$. Note that if the letters of U_i are isolated letters in G_1^i and G_2^i, then they remain as isolated letters in G_1^j and G_2^j. Consider the j-th iteration of Approx-CMSR, and assume Approx-CMSR made an adjacency $a_j \bullet b_j$. Again from Lemma 3 we know that the letters in between a_j and b_j in the original sequences G_1 and G_2 are all in $\cup_{i=1}^{j} U_i$. It follows that none of the letters of U_j resides in a substring of G_1^j and G_2^j that contains a letter $z \notin U_j$, since otherwise z must be in $\cup_{i=1}^{j-1} U_i$, a contradiction to the inductive assumption, or z must be one of a_j and b_j, contradicting to all nine defined operations. Next, similarly as in the base case, one can see that the letters of U_j can form a length-2 substring in G_1^j and G_2^j only if the U_j is from an operation 8. □

Lemma 8 *In the j-th iteration of algorithm Approx-CMSR, for $j = 1, 2, \ldots, m$, assume two letters a_j and b_j are made adjacent into a common substring. If $a_j \in W$, then the adjacencies at a_j in G_1^* and G_2^* are not broken by inserting letters of U_j or b_j into G_1^{j-1} and G_2^{j-1}.*

PROOF. Notice that if any letter of U_j or b_j is in G_1^* and G_2^*, then it is not inserted into G_1^{j-1} and G_2^{j-1}. So the lemma holds when all these letters are kept in G_1^* and G_2^*. Consider first the case $U_j \subset O_1$. If $b_j \in O_1$, inserting b_j into G_1^{j-1} and G_2^{j-1} will not break any adjacency involving a letter in between a_j and b_j in the original sequences G_1 and G_2, since by Lemma 7 such a letter belongs to $\cup_{i=1}^{j-1} U_i$, and thus cannot co-reside with any letter not in between a_j and b_j in the same substring of G_1^{j-1} and G_2^{j-1}. When a_j is adjacent to a letter z, we claim that z resides at the opposite side of a_j with respect to b_j; for otherwise, b_j is closer to a_j than z and thus inserting b_j alone into G_1^* and G_2^* achieves a better than optimum solution, a contradiction. It follows that inserting b_j changes nothing to this adjacency. Afterwards, inserting letters of U_j does not do anything to the adjacencies involving a_j and b_j.

In the other case of $U_j - O_1 \neq \emptyset$, a_j can only be adjacent to a letter z at the opposite side of a_j with respect to b_j. Thus, if $b_j \in O_1$, inserting b_j into G_1^{j-1} and G_2^{j-1} will not break any adjacency. Afterwards, inserting the letter of $U_j \cap O_1$, if any, does not do anything to the adjacencies involving a_j and b_j. □

In the following nine Lemmas 9–17, we show that every isolate of $O_1 \cap (U_j \cup V_j)$ can be put into exactly one of bins B_2, B_3^1, and B_3^2, when the j-th iteration of Approx-CMSR executes an operation 1, 2, 3, 4, 5, 6, 7, 8, or 9, respectively. During this process, we also identify the letters of $V_j - O_1$ that belong to W.

Lemma 9 *If* Approx-CMSR *executes an operation 1 in the j-th iteration, then $O_1 \cap (U_j \cup V_j) \subset B_2$, i.e., every isolate of $O_1 \cap (U_j \cup V_j)$ is attributed with at most 2 isolates of $U \cup R \cup W$.*

PROOF. Recall that an operation 1 deletes isolate x_j, and forms a length-2 type-1 substring $a_j \bullet b_j$, *i.e.*, $U_j = \{x_j\}$ and $V_j = \{a_j, b_j\}$ (see Table 1). From Lemma 3, we know that in the original sequences G_1 and G_2 all letters in between a_j and b_j belong to $\cup_{i=1}^{j} U_i$. We discuss two cases.

In the first case, $x_j \in O_1$. If $a_j, b_j \in O_1$, inserting them into G_1^{j-1} and G_2^{j-1} in the worst case would break two current length-2 substrings, one in each sequence, giving rise to a maximum of 4 isolated letters; If $a_j \in O_1$ but $b_j \notin O_1$, then $b_j \in W$. In this case, inserting a_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters by Lemmas 7 and 8. The case of $a_j \notin O_1$ but $b_j \in O_1$ is analogously discussed. It follows that when $|O_1 \cap V_j| = 2, 1, 0$, respectively, inserting the isolates of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} in the worst case will generate $4, 0, 0$ new isolated letters, respectively. Afterwards, x_j is inserted back, generating only one isolate which is x_j itself. In summary, when $|O_1 \cap V_j| = 2, 1, 0$, respectively, inserting the isolates of $O_1 \cap (U_j \cup V_j)$ into G_1^{j-1} and G_2^{j-1} in the worst case will generate $5, 1, 1$ new isolated letters of $U \cup R \cup W$, respectively, and thus we can attribute them to isolates of $O_1 \cap (U_j \cup V_j)$ to put the isolates of $O_1 \cap (U_j \cup V_j)$ into bin B_2.

(*Remark.* Note that among these generated isolates, x_j is known to be in $U \cup R$. Therefore, when $a_j \in O_1$ but $b_j \notin O_1$, $x_j \in U \cup R$ and $b_j \in W$ cancel out each other. Nevertheless, we do not need to do this to skip counting x_j in, since on average every isolate of $O_1 \cap (U_j \cup V_j)$ is attributed

with less than two isolated letters of $U \cup R \cup W$. Later, in the proof of Lemma 17, we might have to do the cancelling.)

In the other case, $x_j \notin O_1$. From Lemma 7, we know that if x_j is not an isolated letter in G_1^{j-1} and G_2^{j-1}, then inserting the isolates of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} will make x_j an isolated letter; on the other hand, if x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, then inserting a_j, b_j into G_1^{j-1} and G_2^{j-1} will not break into any substring in both sequences, and thus generating no new isolated letters. It follows that when $|O_1 \cap V_j| = 2$, inserting a_j, b_j into G_1^{j-1} and G_2^{j-1} in the worst case would break a length-3 substring to which x_j belongs, giving rise to a maximum of 3 isolated letters. When $|O_1 \cap V_j| = 1$ and without loss of generality $b_j \notin O_1$ (thus $b_j \in W$), since every letter in between x_j and b_j in the original sequences G_1 and G_2 is already an isolate in G_1^{j-1} and G_2^{j-1}, we conclude that inserting a_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring to which x_j belongs, giving rise to a maximum of 2 isolated letters. (*Remark.* Again, a cancelling pair of x_j and b_j can be identified, but we skip it since it is not needed in this case.) In summary, when $x_j \notin O_1$ and $|O_1 \cap V_j| = 2, 1$, respectively, inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} will generate, in the worst case, $3, 2$ new isolated letters of $U \cup R \cup W$, respectively, and thus we can attribute them to isolates of $O_1 \cap V_j$ to put the isolates of $O_1 \cap V_j$ into bin B_2.

This proves the lemma that every isolate of $O_1 \cap (U_j \cup V_j)$ is attributed with at most 2 isolates of $U \cup R \cup W$. □

Lemma 10 *If* Approx-CMSR *executes an operation 2 in the j-th iteration, then $O_1 \cap (U_j \cup V_j) \subset B_2$.*

Proof. Recall that an operation 2 deletes isolate x_j, and forms two length-2 type-1 substrings $a_j \bullet b_j$ and $c_j \bullet d_j$, *i.e.*, $U_j = \{x_j\}$ and $V_j = \{a_j, b_j, c_j, d_j\}$ (see Table 1). From Lemma 3, we know that in the original sequences G_1 and G_2 all letters in between a_j and b_j (and in between c_j and d_j, respectively) belong to $\cup_{i=1}^{j} U_i$. We discuss two cases.

In the first case, $x_j \in O_1$. The discussion on a_j and b_j (c_j and d_j, respectively) is exactly the same as the first case in the proof of Lemma 9. It follows that when $|O_1 \cap V_j| = 4, 3, 2, 1, 0$, respectively, inserting the isolates of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} in the worst case will generate $8, 4, 4, 0, 0$ new isolated letters, respectively. Afterwards, x_j is inserted back, generating only one isolate which is x_j itself. In summary, when $|O_1 \cap V_j| = 4, 3, 2, 1, 0$, respectively, inserting the isolates of $O_1 \cap (U_j \cup V_j)$ into G_1^{j-1} and G_2^{j-1} in the worst case will generate $9, 5, 5, 1, 1$ new isolated letters of $U \cup R \cup W$, respectively, and thus we can attribute them to isolates of $O_1 \cap (U_j \cup V_j)$ to put the isolates of $O_1 \cap (U_j \cup V_j)$ into bin B_2.

In the other case, $x_j \notin O_1$. From Lemma 7, we know that if x_j is not an isolated letter in G_1^{j-1} and G_2^{j-1}, then inserting the isolates of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} will make x_j an isolated letter; on the other hand, if x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, then inserting a_j, b_j (or c_j, d_j, respectively) into G_1^{j-1} and G_2^{j-1} will not break into any substring in the sequence where x_j is in between a_j and b_j (in between c_j and d_j, respectively), and thus in the worst case would break one current length-2 substring, giving rise to a maximum of 2 isolated letters. It follows that when $|O_1 \cap V_j| = 4$, inserting a_j, b_j, c_j, d_j into G_1^{j-1} and G_2^{j-1} in the worst case would break two current length-2 substrings and a length-3 substring to which x_j belongs, giving rise to a maximum of 7 isolated letters. When $|O_1 \cap V_j| = 3$ and without loss of generality $d_j \notin O_1$ (thus $d_j \in W$), since every letter in between x_j and d_j in the original sequences G_1 and G_2 is already an isolate in G_1^{j-1}

and G_2^{j-1}, we conclude that inserting a_j, b_j, c_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring and another length-2 substring to which x_j belongs, giving rise to a maximum of 4 isolated letters. Similarly, when $|O_1 \cap V_j| = 2$ and $b_j, d_j \notin O_1$ (thus $b_j, d_j \in W$), we conclude that inserting a_j, c_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring to which x_j belongs, giving rise to a maximum of 2 isolated letters. When $|O_1 \cap V_j| = 2$ and $c_j, d_j \notin O_1$ (thus $c_j, d_j \in W$), we conclude from Lemma 3 that x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, and thus inserting a_j, b_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring, giving rise to a maximum of 2 isolated letters. The other scenarios of $|O_1 \cap V_j| = 2$ can be symmetrically discussed, and at most two new isolate letters are generated. Therefore, when $|O_1 \cap V_j| = 2$, at most two new isolates of $U \cup R \cup W$ are generated. When $|O_1 \cap V_j| = 1$ and assume without loss of generality $a_j \in O_1$ (which implies $b_j, c_j, d_j \in W$), then similarly we conclude from Lemma 3 that x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, and thus inserting a_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters. In summary, when $x_j \notin O_1$ and $|O_1 \cap V_j| = 4, 3, 2, 1$, respectively, inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} will generate, in the worst case, $7, 4, 2, 0$ new isolated letters of $U \cup R \cup W$, respectively, and thus we can attribute them to isolates of $O_1 \cap V_j$ to put the isolates of $O_1 \cap V_j$ into bin B_2.

This proves the lemma that every isolate of $O_1 \cap (U_j \cup V_j)$ is attributed with at most 2 isolates of $U \cup R \cup W$. □

Lemma 11 *If* Approx-CMSR *executes an operation 3 in the j-th iteration, then $O_1 \cap (U_j \cup V_j) \subset B_2$.*

PROOF. Recall that an operation 3 deletes isolate x_j, and forms one length-2 type-1 substrings $a_j \bullet b_j$ and merges c_j to an existing substring S, *i.e.*, $U_j = \{x_j\}$ and $V_j = \{a_j, b_j, c_j\}$ (see Table 1). For ease of discussion, let d_j denote the ending letter of S to which c_j is made adjacent to. From Lemma 3, we know that in the original sequences G_1 and G_2 all letters in between a_j and b_j (and in between c_j and d_j, respectively) belong to $\cup_{i=1}^{j} U_i$. We discuss two cases.

In the first case, $x_j \in O_1$. The discussion on a_j and b_j is exactly the same as the first case in the proof of Lemma 9. For c_j, since d_j is either a letter in some type-0 substring of the original sequences or a letter in some V_i for $i < j$, we conclude that d_j is already in G_1^{j-1} and G_2^{j-1}. Therefore, this can be deemed as a special case of the first case in the proof of Lemma 9. That is, when $|O_1 \cap V_j| = 3, 2, 1, 0$, respectively, inserting the isolates of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} in the worst case will generate $4, 4, 0, 0$ new isolated letters, respectively. Afterwards, x_j is inserted back, generating only one isolate which is x_j itself. In summary, when $|O_1 \cap V_j| = 3, 2, 1, 0$, respectively, inserting the isolates of $O_1 \cap (U_j \cup V_j)$ into G_1^{j-1} and G_2^{j-1} in the worst case will generate $5, 5, 1, 1$ new isolated letters of $U \cup R \cup W$, respectively, and thus we can attribute them to isolates of $O_1 \cap (U_j \cup V_j)$ to put the isolates of $O_1 \cap (U_j \cup V_j)$ into bin B_2.

In the other case, $x_j \notin O_1$. From Lemma 7, we know that if x_j is not an isolated letter in G_1^{j-1} and G_2^{j-1}, then inserting the isolates of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} will make x_j an isolated letter; on the other hand, if x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, then inserting a_j, b_j (or c_j, respectively) into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring, giving rise to a maximum of 2 isolated letters. Note that d_j is already in G_1^{j-1} and G_2^{j-1}, and thus we may treat this as a special case of the second case in the proof of Lemma 9. That is, when $|O_1 \cap V_j| = 3$, inserting a_j, b_j, c_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current

length-2 substring and another length-2 substring to which x_j belongs, giving rise to a maximum of 4 isolated letters. When $|O_1 \cap V_j| = 2$ and $c_j \notin O_1$ (thus $c_j \in W$), then x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, and thus inserting a_j, b_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring, giving rise to a maximum of 2 isolated letters; when $|O_1 \cap V_j| = 2$ and $b_j \notin O_1$ (thus $b_j \in W$), and if x_j is an isolated letter, then inserting a_j, c_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters; if x_j is not an isolated letter, then inserting a_j, c_j into G_1^{j-1} and G_2^{j-1} will have to make x_j an isolated letter, and thus in the worst case would break a length-2 substring to which x_j belongs, giving rise to a maximum of 2 isolated letters. That is, when $|O_1 \cap V_j| = 2$, at most two isolated letters of $U \cup R \cup W$ are generated. If $|O_1 \cap V_j| = 1$ and $a_j \in O_1$ (which implies $b_j, c_j \in W$), then x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, and thus inserting a_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters; If $|O_1 \cap V_j| = 1$ and $c_j \in O_1$ (which implies $a_j, b_j \in W$), then similarly x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, and thus inserting c_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters. That is, when $|O_1 \cap V_j| = 1$, no new isolated letters of $U \cup R \cup W$ will be generated. In summary, when $x_j \notin O_1$ and $|O_1 \cap V_j| = 3, 2, 1$, respectively, inserting the isolates of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} will generate, in the worst case, $4, 2, 0$ new isolated letters of $U \cup R \cup W$, respectively, and thus we can attribute them to isolates of $O_1 \cap V_j$ to put the isolates of $O_1 \cap V_j$ into bin B_2.

This proves the lemma that every isolate of $O_1 \cap (U_j \cup V_j)$ is attributed with at most 2 isolates of $U \cup R \cup W$. □

Lemma 12 *If* Approx-CMSR *executes an operation 4 in the j-th iteration, then $O_1 \cap (U_j \cup V_j) \subset B_2$.*

Proof. Recall that an operation 4 deletes isolate x_j, appends a_j to an existing substring S_1, and appends c_j to another existing substring S_2, *i.e.*, $U_j = \{x_j\}$ and $V_j = \{a_j, c_j\}$ (see Table 1). For ease of discussion, let b_j denote the ending letter of S_1 to which a_j is made adjacent to, and d_j denote the ending letter of S_2 to which c_j is made adjacent to. From Lemma 3, we know that in the original sequences G_1 and G_2 all letters in between a_j and b_j (and in between c_j and d_j, respectively) belong to $\cup_{i=1}^{j} U_i$. We discuss two cases.

In the first case, $x_j \in O_1$. Since b_j (d_j, respectively) is either a letter in some type-0 substring of the original sequences or a letter in some V_i for $i < j$, we conclude that b_j (d_j, respectively) is already in G_1^{j-1} and G_2^{j-1}. Therefore, this can be deemed as a special case of the first case in the proof of Lemma 9. That is, inserting the isolates of $O_1 \cap V_j$, if any, into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters. Afterwards, x_j is inserted back, generating only one isolate which is x_j itself. Therefore, every isolate of $O_1 \cap (U_j \cup V_j)$ belongs to bin B_2.

In the other case, $x_j \notin O_1$. By Lemma 7, inserting a_j and/or c_j into G_1^{j-1} and G_2^{j-1} will have to make x_j an isolated letter, if x_j is not an isolated letter in G_1^{j-1} and G_2^{j-1}; on the other hand, if x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, then inserting a_j and/or c_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters. Therefore, when $|O_1 \cap V_j| = 2$, inserting a_j, c_j into G_1^{j-1} and G_2^{j-1} in the worst case would break a length-2 substring to which x_j belongs, giving rise to a maximum of 2 isolated letters. If $|O_1 \cap V_j| = 1$ and assume without loss of generality that $a_j \in O_1$ (which implies $c_j \in W$), then x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, and thus inserting a_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters. In summary, when $x_j \notin O_1$ and $|O_1 \cap V_j| = 2, 1$, respectively, inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} will generate, in

the worst case, $2, 0$ isolated letters, respectively. Therefore, we can arbitrarily attribute them to isolates of $O_1 \cap V_j$ to put the isolates of $O_1 \cap V_j$ into bin B_2. □

Lemma 13 *If* Approx-CMSR *executes an operation 5 in the j-th iteration, then every isolate of $O_1 \cap (U_j \cup V_j)$ is either in B_2, or in B_3^1.*

Proof. Recall that bin B_3^1 contains letters of OPT each of which is attributed with at most three isolates of $U \cup R \cup W$ and is associated with a unique isolate copy of $O_1 \cap R$. Also, an operation 5 deletes isolate x_j, and appends a_j to an existing substring S, *i.e.*, $U_j = \{x_j\}$ and $V_j = \{a_j\}$ (see Table 1). For ease of discussion, let b_j denote the ending letter of S to which a_j is made adjacent to. From Lemma 3, we know that in the original sequences G_1 and G_2 all letters in between a_j and b_j belong to $\cup_{i=1}^{j} U_i$. We discuss two cases.

In the first case, $x_j \in O_1$. If $a_j \in O_1$, inserting a_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters, similarly proven as in Lemma 8. Hence, $a_j \in B_2$. Afterwards, x_j is inserted back, and it could break an adjacency in G_1^{j-1} and G_2^{j-1}, denoted as $c_j \bullet d_j$. If one of c_j and d_j does not belong to $U \cup R \cup V$ (which is a superset of $U \cup R \cup W$), we conclude that $x_j \in B_2$; otherwise, due to the fact that this is an operation 5, there must be some other isolate from $\cup_{i=j+1}^{m} (U_i \cup V_i)$ or from R that resides in between c_j and d_j is the original sequences G_1 and G_2. In the former case, c_j and d_j are not attributed to x_j by the attribution rule we set earlier, and thus $x_j \in B_2$; in the latter, select an arbitrary isolate from R, denoted as x_j^R, that resides in between c_j and d_j in one of the original sequences G_1 and G_2, and associate this copy of x_j^R with x_j. This way, c_j, d_j, x_j are attributed to x_j and therefore $x_j \in B_3^1$. It is important to notice that since the adjacency between c_j and d_j is broken by x_j, no more copies of isolates of $O_1 \cap R$ residing in between c_j and d_j in the corresponding original sequence would be associated with other isolates of O_1. That is, this isolate copy x_j^R is unique to x_j.

In the other case, $x_j \notin O_1$. By Lemma 7, inserting a_j into G_1^{j-1} and G_2^{j-1} will have to make x_j an isolated letter, if x_j is not an isolated letter in G_1^{j-1} and G_2^{j-1}; on the other hand, if x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, then inserting a_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters. Note that by Lemma 7 if x_j is not an isolated letter in G_1^{j-1} and G_2^{j-1} then we must have $a_j \in O_1$; and inserting a_j into G_1^{j-1} and G_2^{j-1} in the worst case would break a length-2 substring to which x_j belongs, giving rise to a maximum of 2 isolated letters. Hence, $a_j \in B_2$. □

Lemma 14 *If* Approx-CMSR *executes an operation 6 in the j-th iteration, then $O_1 \cap (U_j \cup V_j) \subset B_2$.*

Proof. Recall that an operation 6 deletes isolates x_j, y_j, and forms a novel length-3 substring $a_j \bullet b_j \bullet c_j$ and a novel length-2 substring $d_j \bullet e_j$, *i.e.*, $U_j = \{x_j, y_j\}$ and $V_j = \{a_j, b_j, c_j, d_j, e_j\}$ (see Table 1). From Lemma 3, we know that in the original sequences G_1 and G_2 all letters in between a_j and b_j (in between b_j and c_j, and in between d_j and e_j) belong to $\cup_{i=1}^{j} U_i$. We discuss three cases.

In the first case, $x_j, y_j \in O_1$. If $|O_1 \cap V_j| = 5$, by Lemma 3, inserting a_j, b_j, c_j into G_1^{j-1} and G_2^{j-1} in the worst case would break two current length-2 substrings, giving rise to a maximum of 4

isolated letters; inserting d_j, e_j into G_1^{j-1} and G_2^{j-1} in the worst case would also break two current length-2 substrings, giving rise to a maximum of 4 isolated letters. That is, in this scenario there are a maximum of 8 new isolated letters generated. If $d_j \in O_1$ but $e_j \notin O_1$ (thus $e_j \in W$), by Lemma 3, inserting d_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters other than the isolated letters generated by inserting a_j, b_j, c_j, which in the worst case would break two current length-2 substrings giving rise to a maximum of 4 isolated letters. Symmetric discussions apply to a_j, b_j, c_j and d_j. Therefore, if $|O_1 \cap V_j| = 4, 3, 2$, inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} in the worst case would break two current length-2 substrings, giving rise to a maximum of 4 isolated letters. If $|O_1 \cap V_j| = 1$ and assume without loss of generality that $a_j \in O_1$ (which implies $b_j \in W$), inserting a_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters. In summary, when $x_j, y_j \in O_1$ and $|O_1 \cap V_j| = 5, 4, 3, 2, 1, 0$, respectively, inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} will generate, in the worst case, $8, 4, 4, 4, 0, 0$ isolated letters, respectively. Afterwards, x_j, y_j are inserted back, generating only two isolates which are x_j, y_j themselves. Therefore, we can attribute them to isolates of $O_1 \cap (U_j \cup V_j)$ to put the isolates of $O_1 \cap (U_j \cup V_j)$ into bin B_2.

In the second case, $x_j \notin O_1$ and $y_j \in O_1$ (the symmetric case $x_j \in O_1$ and $y_j \notin O_1$ can be similarly discussed). Note that if x_j is not an isolated letter in G_1^{j-1} and G_2^{j-1}, then inserting the units of $O_1 \cap V_j$ has to make it singleton; while if x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, by Lemma 3, then inserting a_j, b_j, c_j (d_j, e_j, respectively) would break at most one current length-2 substring to give rise to two new isolated letters. It follows that, if $|O_1 \cap V_j| = 5$, inserting a_j, b_j, c_j, d_j, e_j into G_1^{j-1} and G_2^{j-1} in the worst case would break two current length-2 substrings and one current length-3 substring to which x_j belongs, giving rise to a maximum of 7 isolated letters. If $d_j \in O_1$ but $e_j \notin O_1$ (thus $e_j \in W$), by Lemma 3, inserting d_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters other than the isolated letters generated by inserting a_j, b_j, c_j, which in the worst case would break one current length-2 substring and another current length-2 substring to which x_j belongs, giving rise to a maximum of 4 isolated letters. Symmetric discussions apply to a_j, b_j, c_j and d_j. Therefore, if $|O_1 \cap V_j| = 4, 3, 2$, inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} in the worst case would break two current length-2 substrings, giving rise to a maximum of 4 isolated letters. If $|O_1 \cap V_j| = 1$ and assume without loss of generality that $a_j \in O_1$ (which implies $b_j \in W$), then x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, and thus inserting a_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters. In summary, when exactly one of x_j, y_j is in O_1 and $|O_1 \cap V_j| = 5, 4, 3, 2, 1, 0$, respectively, inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} will generate, in the worst case, $7, 4, 4, 4, 0, 0$ isolated letters, respectively. Afterwards, y_j is inserted back, generating only one isolate which is y_j itself. Therefore, we can attribute them to isolates of $O_1 \cap (U_j \cup V_j)$ to put the isolates of $O_1 \cap (U_j \cup V_j)$ into bin B_2.

In the last case, $x_j, y_j \notin O_1$. Note that if x_j (y_j, respectively) is not an isolated letter in G_1^{j-1} and G_2^{j-1}, then inserting the units of $O_1 \cap V_j$ has to make it singleton; while if x_j (y_j, respectively) is already an isolated letter in G_1^{j-1} and G_2^{j-1}, by Lemma 3, then inserting the units of $O_1 \cap \{a_j, b_j\}$ or the units of $O_1 \cap \{d_j, e_j\}$ (the units of $O_1 \cap \{b_j, c_j\}$ or the units of $O_1 \cap \{d_j, e_j\}$, respectively) would break at most one current length-2 substring. It follows that, if $|O_1 \cap V_j| = 5$, inserting a_j, b_j, c_j, d_j, e_j into G_1^{j-1} and G_2^{j-1} in the worst case would break two current length-2 substrings and one current length-4 substring to which x_j, y_j belong, giving rise to a maximum of 8 isolated letters. If $d_j \in O_1$ but $e_j \notin O_1$ (thus $e_j \in W$), by Lemma 3, inserting d_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters other than the isolated letters generated by inserting a_j, b_j, c_j, which in the worst case would break one current length-2 substring and another current length-3 substring to which x_j, y_j belong, giving rise to a maximum of 5 isolated letters. Symmetric

discussions apply to a_j, b_j, c_j and d_j. Therefore, if $|O_1 \cap V_j| = 4, 3$, inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring and one current length-3 substring, giving rise to a maximum of 5 isolated letters. When $|O_1 \cap V_j| = 2$ and $a_j, b_j \in O_1$ (which imply $c_j, d_j, e_j \in W$), if x_j is not an isolated letter in G_1^{j-1} and G_2^{j-1}, then neither is y_j, and $x_j \bullet y_j$ is a length-2 substring in G_1^{j-1} and G_2^{j-1}. The argument in the last paragraph still applies, but inserting a_j, b_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring $x_j \bullet y_j$, giving rise to a maximum of 2 isolated letters. Similar discussion applies to other pairs such as $d_j, e_j \in O_1$. When $|O_1 \cap V_j| = 2$ and $a_j, d_j \in O_1$ (which imply $b_j, c_j, e_j \in W$), we conclude that y_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, and thus similarly inserting a_j, d_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring to which x_j belongs, giving rise to a maximum of 2 isolated letters. That is, if $|O_1 \cap V_j| = 2$, then a maximum of 2 isolated letters can be generated. When $|O_1 \cap V_j| = 1$ and $a_j \in O_1$ (similar discussions apply to c_j, d_j, e_j), then both x_j and y_j are already isolated letters in G_1^{j-1} and G_2^{j-1}, and thus inserting a_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters. When $|O_1 \cap V_j| = 1$ and $b_j \in O_1$, if x_j is not an isolated letter in G_1^{j-1} and G_2^{j-1}, then neither is y_j, and $x_j \bullet y_j$ is a length-2 substring in G_1^{j-1} and G_2^{j-1}. The argument in the last paragraph still applies, but inserting b_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring $x_j \bullet y_j$, giving rise to a maximum of 2 isolated letters. In summary, when $x_j, y_j \notin O_1$ and $|O_1 \cap V_j| = 5, 4, 3, 2, 1$, respectively, inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} will generate, in the worst case, $8, 5, 5, 2, 2$ isolated letters, respectively. Therefore, we can arbitrarily attribute them to isolates of $O_1 \cap V_j$ to put the isolates of $O_1 \cap V_j$ into bin B_2.

This proves the lemma that every isolate of $O_1 \cap (U_j \cup V_j)$ is attributed with at most 2 isolates of $U \cup R \cup W$, thus belonging to B_2. □

Lemma 15 *If* Approx-CMSR *executes an operation 7 in the j-th iteration, then every isolate of $O_1 \cap (U_j \cup V_j)$ is either in B_2, or in B_3^1.*

PROOF. Recall that bin B_3^1 contains letters of OPT each of which is attributed at most three isolates of $U \cup R \cup W$ and is associated with a unique isolate copy of $O_1 \cap R$. Also, an operation 7 deletes isolate x_j, and forms a novel length-2 substring $a_j \bullet b_j$, *i.e.*, $U_j = \{x_j\}$ and $V_j = \{a_j, b_j\}$ (see Table 1). From Lemma 3, we know that in the original sequences G_1 and G_2 all letters in between a_j and b_j belong to $\cup_{i=1}^{j} U_i$. We discuss two cases.

In the first case, $x_j \in O_1$. If $|O_1 \cap V_j| = 2$, by Lemma 3, inserting a_j, b_j into G_1^{j-1} and G_2^{j-1} in the worst case would break two current length-2 substrings, giving rise to a maximum of 4 isolated letters. If $|O_1 \cap V_j| = 1$ and assume without loss of generality that $a_j \in O_1$ (thus $b_j \in W$), by Lemma 7, inserting a_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters. In summary, when $x_j \in O_1$ and $|O_1 \cap V_j| = 2, 1$, respectively, inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} will generate, in the worst case, $4, 0$ isolated letters, respectively. Hence, $O_1 \cap V_j \subset B_2$. Afterwards, x_j is inserted back, and it could break an adjacency in G_1^{j-1} and G_2^{j-1}, denoted as $c_j \bullet d_j$. If one of c_j and d_j does not belong to $U \cup R \cup V$, we conclude that $x_j \in B_2$; otherwise, due to the fact that this is an operation 7, there must be some other isolate from $\cup_{i=j+1}^{m} (U_i \cup V_i)$ or from R that resides in between c_j and d_j in the original sequences G_1 and G_2. In the former case, c_j and d_j are not attributed to x_j by the attribution rule we set up, and thus $x_j \in B_2$; in the latter, select an arbitrary isolate from R, denoted as x_j^R, that resides in between c_j and d_j in one of the original

sequences G_1 and G_2, and associate this copy of x_j^R with x_j. This way, c_j, d_j, x_j are attributed to x_j and therefore $x_j \in B_3^1$. Again, similarly as in the proof of Lemma 13, it is important to notice that since the adjacency between c_j and d_j is broken by x_j, no more copies of isolates of $O_1 \cap R$ residing in between c_j and d_j in the corresponding original sequence would be associated with other isolates of O_1. That is, this isolate copy x_j^R is unique to x_j.

In the other case, $x_j \notin O_1$ (which implies $V_j - O_1 \subset W$). By Lemma 7, inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} has to make x_j an isolated letter, if x_j is not an isolated letter in G_1^{j-1} and G_2^{j-1}; on the other hand, if x_j is an isolated letter in G_1^{j-1} and G_2^{j-1}, then inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring, giving rise to a maximum of 2 isolated letters. Therefore, if $|O_1 \cap V_j| = 2$, inserting a_j, b_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring, denoted as $r_j \bullet s_j$, and one current length-3 substring to which x_j belongs, denoted as $t_j \bullet x_j \bullet u_j$, giving rise to a maximum of 5 isolated letters r_j, s_j, t_j, x_j, u_j. If one of these five is not an isolate of $U \cup R \cup V$, then a maximum of four new isolates are generated, and thus $a_j, b_j \in B_2$. In the other case, by Lemma 7 they all belong to $\left(\cup_{i=j}^{m} U_i\right) \cup R \cup V$. Since in the j-th iteration Approx-CMSR goes with a lower priority operation 7 than an operation 6, we conclude that there must be some other isolate from $\cup_{i=j+1}^{m} U_i$ or from R that resides in between at least one pair of $\{r_j, a_j\}$, $\{b_j, s_j\}$, $\{t_j, a_j\}$, and $\{b_j, u_j\}$ in the original sequences G_1 and G_2. In the former case, at least one of r_j, s_j, t_j, u_j is not attributed to a_j, b_j by the attribution rule we set up, and thus $a_j, b_j \in B_2$; in the latter and assume a letter of R residing in between r_j and a_j in one of the original sequences G_1 and G_2, select an arbitrary such isolate from R, denoted as a_j^R, and associate this copy of a_j^R with a_j. This way, r_j, t_j, x_j are attributed to a_j and therefore $a_j \in B_3^1$; s_j, u_j are attributed to b_j and therefore $b_j \in B_2$. Again, similarly as in the proof of Lemma 13, it is important to notice that since the adjacency between r_j and s_j is broken by a_j, b_j, no more copies of isolates of $O_1 \cap R$ residing in between r_j and s_j in the corresponding original sequence would be associated with other isolates of O_1. That is, this isolate copy a_j^R is unique to a_j. If $|O_1 \cap V_j| = 1$ and assume $a_j \in O_1$, inserting a_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring to which x_j belongs, giving rise to a maximum of 2 isolated letters, and thus $a_j \in B_2$. This proves the lemma. □

Lemma 16 *If* Approx-CMSR *executes an operation 8 in the j-th iteration, then every isolate of $O_1 \cap (U_j \cup V_j)$ is either in B_2, or in B_3^1.*

Proof. Recall that bin B_3^1 contains letters of OPT each of which is attributed at most three isolates of $U \cup R \cup W$ and is associated with a unique isolate copy of $O_1 \cap R$. Also, an operation 8 deletes isolates x_j, y_j, and forms a novel length-2 substring $a_j \bullet b_j$, *i.e.*, $U_j = \{x_j, y_j\}$ and $V_j = \{a_j, b_j\}$ (see Table 1). From Lemma 3, we know that in the original sequences G_1 and G_2 all letters in between a_j and b_j belong to $\cup_{i=1}^{j} U_i$. We discuss three cases.

In the first case, $x_j, y_j \in O_1$. If $|O_1 \cap V_j| = 2$, by Lemma 3, inserting a_j, b_j into G_1^{j-1} and G_2^{j-1} in the worst case would break two current length-2 substrings, giving rise to a maximum of 4 isolated letters. If $|O_1 \cap V_j| = 1$ and assume $a_j \in O_1$ (thus $b_j \in W$), by Lemma 7, inserting a_j into G_1^{j-1} and G_2^{j-1} will not generate any new isolated letters. In summary, when $x_j, y_j \in O_1$ and $|O_1 \cap V_j| = 2, 1$, respectively, inserting the units of $O_1 \cap V_j$ into G_1^{j-1} and G_2^{j-1} will generate, in the worst case, $4, 0$ isolated letters, respectively. Therefore, we can arbitrarily attribute them to isolates of $O_1 \cap V_j$ to put the isolates of $O_1 \cap V_j$ into bin B_2. Afterwards, x_j, y_j are inserted

back, and they could break two adjacencies in G_1^{j-1} and G_2^{j-1}, denoted as $c_j \bullet d_j$ broken by x_j and $e_j \bullet f_j$ broken by y_j. If one of c_j and d_j does not belong to $U \cup R \cup V$, we conclude that $x_j \in B_2$; otherwise, due to the fact that this is an operation 8 of priority lower than an operation 7, there must be some other isolate from $\cup_{i=j+1}^{m}(U_i \cup V_i)$ or from R that resides in between c_j and d_j in the original sequences G_1 and G_2. In the former case, c_j and d_j are not attributed to x_j by the attribution rule we set up earlier, and thus $x_j \in B_2$; in the latter, select an arbitrary isolate from R, denoted as x_j^R, that resides in between c_j and d_j in one of the original sequences G_1 and G_2, and associate this copy of x_j^R with x_j. This way, c_j, d_j, x_j are attributed to x_j and therefore $x_j \in B_3^1$. Again, similarly as in the proof of Lemma 13, it is important to notice that since the adjacency between c_j and d_j is broken by x_j, no more copies of isolates of $O_1 \cap R$ residing in between c_j and d_j in the corresponding original sequence would be associated with other isolates of O_1. That is, this isolate copy x_j^R is unique to x_j. Analogous argument applies to e_j, f_j, y_j.

In the second case, $x_j \notin O_1$ and $y_j \in O_1$ (the symmetric case $x_j \in O_1$ and $y_j \notin O_1$ can be similarly discussed). Note that if x_j is not an isolated letter in G_1^{j-1} and G_2^{j-1}, then inserting the units of $O_1 \cap V_j$ has to make it singleton; while if x_j is already an isolated letter in G_1^{j-1} and G_2^{j-1}, by Lemma 3, then inserting a_j, b_j would break at most one current length-2 substring to give rise to two isolated letters. It follows that, if $|O_1 \cap V_j| = 2$, inserting a_j, b_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring, denoted as $r_j \bullet s_j$, and one current length-3 substring to which x_j belongs, denoted as $t_j \bullet x_j \bullet u_j$, giving rise to a maximum of 5 isolated letters r_j, s_j, t_j, x_j, u_j. A similar argument as in the proof of Lemma 15, the second case where $x_j \notin O_1$, shows that $b_j \in B_2$, and either $a_j \in B_2$ too or $a_j \in B_3^1$ and it is associated with a unique isolate copy a_j^R of $O_1 \cap R$ that resides in between t_j and x_j in the original sequences G_1 and G_2. If $|O_1 \cap V_j| = 1$ and assume $a_j \in O_1$ (thus $b_j \in W$), inserting a_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring to which x_j belongs, denoted as $t_j \bullet x_j$, giving rise to a maximum of 2 isolated letters. Thus $a_j \in B_2$. Afterwards, y_j is inserted back, and it could break one adjacency in G_1^{j-1} and G_2^{j-1}, denoted as $c_j \bullet d_j$. Then, a similar argument as in the last paragraph for c_j, d_j, x_j shows that either $y_j \in B_2$ or $y_j \in B_3^1$ and it is associated with a unique isolate copy y_j^R of $O_1 \cap R$ that resides in between c_j and d_j in the original sequences G_1 and G_2.

In the last case, $x_j, y_j \notin O_1$ (which implies $V_j - O_1 \subset W$). If $|O_1 \cap V_j| = 2$, inserting a_j, b_j into G_1^{j-1} and G_2^{j-1} in the worst case would break one current length-2 substring not involving x_j or y_j. We attribute one new isolated letter to each of a_j and b_j. If inserting a_j into G_1^{j-1} and G_2^{j-1} breaks an adjacency $t_j \bullet x_j$ and $t_j \in U \cup R \cup W$, then there must be some other isolate from $\cup_{i=j+1}^{m}(U_i \cup V_i)$ or from R that resides in between t_j and x_j in the original sequences G_1 and G_2, since otherwise a higher priority operation 7 should be performed in the j-th iteration of Approx-CMSR. In the former case, t_j is not attributed to a_j, and thus $a_j \in B_2$; in the latter, select an arbitrary isolate from R, denoted as a_j^R, that resides in between t_j and x_j in one of the original sequences G_1 and G_2, and associate this copy of a_j^R with a_j. This way, a_j is attributed with three isolated letters and therefore $a_j \in B_3^1$. Similar discussions apply for b_j, if $b_j \in O_1$. This proves the lemma.

Note that in G_1^j and G_2^j, none of x_j and y_j can participate a substring containing another letter, but $x_j \bullet y_j$ can be a substring of G_1^j and G_2^j (Lemma 7), while in this case a_j, b_j are either in B_2 or in W — Corollary 1 — a fact to be used in the proof of Theorem 1. □

Corollary 1 *If U_j is resulted from an operation 8 and $x_j \bullet y_j$ is a common substring of G_1^j and G_2^j, then each letter of V_j is either in B_2 or in W.*

Lemma 17 *If* Approx-CMSR *executes an operation 9 in the j-th iteration, then every isolate of $O_1 \cap (U_j \cup V_j)$ is either in B_2, or in B_3^1, or in B_3^2.*

Proof. Recall that bin B_3^1 contains letters of OPT each of which is attributed at most three isolates of $U \cup R \cup W$ and is associated with a unique isolate copy of $O_1 \cap R$; bin B_3^2 contains letters of OPT each of which is attributed at most three isolates of $U \cup R \cup W$ and is associated with two unique isolate copies of $O_1 \cap R$. Also, an operation 9 deletes isolates x_j, y_j, and forms a novel length-2 substring $a_j \bullet b_j$, *i.e.*, $U_j = \{x_j, y_j\}$ and $V_j = \{a_j, b_j\}$ (see Table 1). Here x_j is the only letter separating a_j and b_j in one of the two sequences, while y_j is the only letter separating a_j and b_j in the other sequence. From Lemma 3, we know that in the original sequences G_1 and G_2 all letters in between a_j and b_j belong to $\cup_{i=1}^{j} U_i$. We discuss three cases.

The first case of $x_j, y_j \in O_1$ and the second case of $x_j \notin O_1$ and $y_j \in O_1$ can be identically argued as the first two cases in the proof of Lemma 16, with the conclusion that every isolate of $O_1 \cap (U_j \cup V_j)$ is in $B_2 \cup B_3^1$.

In the last case, $x_j, y_j \notin O_1$ (which implies $V_j - O_1 \subset W$). If $a_j \in O_1$ and inserting a_j into G_1^{j-1} and G_2^{j-1} breaks an adjacency, then this adjacency must involve either x_j or y_j. Assume the inserting breaks an adjacency $t_j \bullet x_j$ where $t_j \in U \cup R \cup V$, there must be some other isolate from $\cup_{i=j+1}^{m} (U_i \cup V_i)$ or from R that resides in between t_j and x_j in the original sequences G_1 and G_2, since otherwise a higher priority operation 7 should be performed in the j-th iteration of Approx-CMSR. In the former case, t_j is not attributed to a_j; in the latter, t_j, x_j are attributed to a_j, and select an arbitrary isolate from R, denoted as a_j^R, that resides in between t_j and x_j in one of the original sequences G_1 and G_2, and associate this copy of a_j^R with a_j. If inserting a_j into G_1^{j-1} and G_2^{j-1} breaks another adjacency $r_j \bullet y_j$ where $r_j \in U \cup R \cup V$, then again there must be some other isolate from $\cup_{i=j+1}^{m} (U_i \cup V_i)$ or from R that resides in between r_j and y_j in the original sequences G_1 and G_2, since otherwise at the j-th iteration Approx-CMSR would execute an operation 7 to delete a_j. In the former case, r_j is not attributed to a_j, and thus $a_j \in B_2$; in the latter, r_j is attributed to a_j, and select an arbitrary isolate from R, denoted as $\hat{a}_j^R$, that resides in between r_j and y_j in one of the original sequences G_1 and G_2, and associate this copy of $\hat{a}_j^R$ with a_j. In summary, either $a_j \in B_2$, or $a_j \in B_3^2$ and it is associated with two unique isolate copies a_j^R and $\hat{a}_j^R$ of $O_1 \cap R$ that resides in between t_j and x_j and in between r_j and y_j, respectively, in the original sequences G_1 and G_2.

Note that if $b_j \in O_1$, then the same argument applies and y_j is attributed to b_j, thus putting b_j either in B_2 or in B_3^2 and associated with two unique isolate copies b_j^R and $\hat{b}_j^R$ of $O_1 \cap R$. If $b_j \notin O_1$, then $b_j \in W$. Furthermore, when $a_j \in B_2$, we conclude that y_j is already an isolated letter. In the other case that $a_j \in B_3^2$, y_j is newly generated isolated letter; however, $y_j \in U$ is not attributed to a_j but it and $b_j \in W$ cancel out each other, meaning that if b_j is attributed to some letter of OPT, it is replaced by y_j, or if the adjacency involving b_j is not broken after inserting all letters of $O_1 \cap (U \cup V)$, then at least two isolate copies of $O_1 \cap R$ are needed to break this adjacency (due to its lowest priority), to these isolate copies y_j is attributed. To summarize, whenever such a canceling pair occurs, $a_j \in B_3^2$ — Corollary 2 — a fact to be used in the proof of Theorem 1. □

Corollary 2 *If U_j is resulted from an operation 9 and one letter of U_j and one letter of V_j form a canceling pair, then the other letter of V_j must be in B_3^2.*

Denote $O_1' = O_1 \cap R$. Note that at the end of the m-th iteration in the algorithm Approx-CMSR, deleting two isolate copies is impossible to form a novel type-1 substring, neither deleting one isolate copy is possible to append an isolate to an existing substring. That is, for any two remaining isolates $r_1, r_2 \in R$ that are not separated by any existing (type-0 or type-1) substring in both sequences, there are at least three other isolate copies in between them, in two sequences; for any remaining isolate $r_1 \in R$, there are at least two other isolate copies in between it and its neighboring substring, in two sequences. We have the following lemma.

Lemma 18 *In G_1^m and G_2^m,*

1) *if there is an adjacency $r_1 \bullet r_2$ for $r_1, r_2 \in R$, then r_1 and r_2 are separated by at least three isolate copies of O_1';*

2) *if there is an adjacency $r \bullet a$ for $r \in R$ and $a \in \Sigma - U - R$, then r and a are separated by at least two isolates of O_1';*

3) *if there is an adjacency $x_j \bullet y_j$ for $x_j, y_j \in U_j$ from some operation 8, then x_j and y_j are separated by at least two isolate copies of O_1';*

4) *if there is an adjacency $v_j \bullet a$ for $v_j \in V_j$ from some operation 8 or 9 and $a \in \Sigma - R$, then v and a are separated by at least two isolate copies of O_1'.*

PROOF. Items 1) and 2) have been established in the last paragraph.

For items 3) and 4), if there is only one isolate copy of O_1' in between the two letters under consideration, then we conclude that in the j-th iteration of algorithm Approx-CMSR, a higher priority operation can be executed, since an operation 8 or 9 deletes at least two isolate copies of O_1', a contradiction. □

In the sequences G_1^m and G_2^m obtained after inserting units of $O_3 \cup O_2 \cup (O_1 - O_1')$ into G_1^* and G_2^*, some units of R are already isolates, while the other still reside in substrings (of length at least two). From Lemmas 9–17, we know that all units of $U \cup V$ are in G_1^m and G_2^m. From Lemma 7 we know that if an $x_j \in U_j$ is not an isolate in G_1^m and G_2^m, then $U_j = \{x_j, y_j\}$ is resulted from an operation 8 and $x_j \bullet y_j$ is a length-2 substring in G_1^m and G_2^m; furthermore, from Corollary 1 we know that the corresponding $V_j = \{a_j, b_j\} \subset B_2 \cup W$. Also, from Corollary 2 we know that when a canceling pair (y_j, b_j) is identified, $b_j \in W$ and the corresponding $a_j \in B_3^2$. After inserting all the isolates of O_1' back into G_1^m and G_2^m, we will obtain the two original sequences G_1 and G_2 and thus all letters of $U \cup V \cup R$, if still residing in substrings of G_1^m and G_2^m, have to be singled out.

Let $S_1, S_2, \ldots, S_k$ denote the substrings in G_1^m and G_2^m that are purely made of isolates of R, and $T_1, T_2, \ldots, T_\ell$ denote the fragments of substrings in G_1^m and G_2^m, where the substrings are not purely made of isolates of R, but the fragments are. Note that $|S_i| \geq 2$ for every $i = 1, 2, \ldots, k$. Let h denote the number of length-2 substrings $x_j \bullet y_j$ in G_1^m and G_2^m, where $x_j, y_j \in U_j$ are resulted from an operation 8; let g denote the number of letters of $v_j \in V_j$ that is still adjacent to another

letter not in R, where V_j is resulted from an operation 8 or 9; and let p denote the number of canceling pairs identified in Lemma 17 (and Corollary 2).

Lemma 19 $2|O'_1| \geq 1.5\sum_{i=1}^{k}|S_i| + 2\sum_{j=1}^{\ell}|T_j| + 2h + |B_3^1| + 2|B_3^2| + g$.

Proof. From Lemma 18, at least two isolate copies of O'_1 are needed to chop off each T_j, for $j = 1, 2, \ldots, \ell$; afterwards, at least three isolate copies of O'_1 are needed to break each adjacency in S_i's and T_j's. These together sum to $2\ell + \sum_{i=1}^{k} 3(|S_i|-1) + \sum_{j=1}^{\ell} 3(|T_j|-1) \geq 1.5\sum_{i=1}^{k}|S_i| + 2\sum_{j=1}^{\ell}|T_j|$.

For each length-2 substrings $x_j \bullet y_j$ in G_1^m and G_2^m, where $x_j, y_j \in U_j$ are resulted from an operation 8, at least two isolate copies of O'_1 are needed to break the adjacency. In addition, for the adjacencies already been broken by inserting units of $O_1 - O'_1$ into G_1^0 and G_1^0, we need at least one unique isolate copy of O'_1 for each unit in B_3^1 and at least two unique isolate copies of O'_1 for each unit in B_3^2. For each letter $v_j \in V_j$ that is still adjacent to some other letter $\Sigma - R$, where V_j is resulted from an operation 8 or 9, at least two isolate copies of O'_1 is needed to break the adjacency at v_j; because in the extreme case v_j is adjacent to another letter of the same membership, we need at least g isolate copies of O'_1.

Note that each of the above isolate copies of O'_1 cannot be used for two purposes. Because each isolate of O'_1 has exactly two copies, we have

$$2|O'_1| \geq 1.5\sum_{i=1}^{k}|S_i| + 2\sum_{j=1}^{\ell}|T_j| + 2h + |B_3^1| + 2|B_3^2| + g, \tag{3.1}$$

where $|S_i|$ and $|T_j|$ denote the length of substrings S_i and T_j respectively, and $|O'_1|$ denotes the number of isolates in O'_1. This proves the lemma. □

Theorem 1 *The Approx-CMSR is a $\frac{7}{3}$-approximation algorithm for the CMSR problem.*

Proof. Approx-CMSR runs in polynomial time in n, where n is the number of letters in the input sequences, since in each iteration of the second step the deleting-retaining operation can be determined in $O(n^2)$ time and there are $O(n)$ iterations.

Note that some of the isolates of $U \cup R$ are attributed to letters of $OPT - O'_1$, two to each letter of B_2 and three to each letter of $B_3^1 \cup B_3^2$. There are at most $2|B_2| + 3|B_3^1| + 3|B_3^2|$ such isolates. Every other isolate of $U \cup R$ belongs to either O'_1, or one of the length-2 substrings $x_j \bullet y_j$ in G_1^m and G_2^m, or one of S_i or T_j, or is cancelled out with a letter of W. For each canceling pair (y_j, b_j), if b_j is isolated in G_1^m and G_2^m, then y_j has been attributed to some letter of $B_2 \cup B_3^1 \cup B_3^2$ and thus has been counted; otherwise, we need to count y_j in. Let p' denote the number of canceling pairs for which the associated b_j is in some substring of G_1^m and G_2^m. It follows that there are at most $|O'_1| + \sum_{i=1}^{k}|S_i| + \sum_{j=1}^{\ell}|T_j| + 2h + p'$ other isolates of $U \cup R$. Hence,

$$|U \cup R| \leq 2|B_2| + 3|B_3^1| + 3|B_3^2| + |O'_1| + \sum_{i=1}^{k}|S_i| + \sum_{j=1}^{\ell}|T_j| + 2h + p'. \tag{3.2}$$

Since for each length-2 substring $x_j \bullet y_j$ in G_1^m and G_2^m, the two corresponding letters of V_j are either in B_2 or in W (Corollary 1), we conclude that $2h + p' \leq |B_2| + g$. Also, since for each canceling pair

(y_j, b_j) the corresponding letter a_j is in B_3^2 (Corollary 2), we conclude that $p' \leq |B_3^2|$. It follows from Equations (3.1) and (3.2) that

$$\begin{aligned}
|U \cup R| &\leq 2|B_2| + 3|B_3^1| + 3|B_3^2| + |O_1'| + \textstyle\sum_{i=1}^{k} |S_i| + \sum_{j=1}^{\ell} |T_j| + 2h + p' \\
&\leq 2|B_2| + 3|B_3^1| + 3|B_3^2| + |O_1'| + \textstyle\sum_{i=1}^{k} |S_i| + \sum_{j=1}^{\ell} |T_j| + \frac{4}{3}h + \frac{1}{3}(|B_2| + g) + \frac{2}{3}|B_3^2| \\
&= \tfrac{7}{3}|B_2| + \tfrac{7}{3}|B_3^1| + \tfrac{7}{3}|B_3^2| + |O_1'| + \Big(\textstyle\sum_{i=1}^{k} |S_i| + \sum_{j=1}^{\ell} |T_j| + \frac{4}{3}h + \frac{2}{3}|B_3^1| + \frac{4}{3}|B_3^2| + \frac{1}{3}g\Big) \\
&\leq \tfrac{7}{3}|B_2| + \tfrac{7}{3}|B_3^1| + \tfrac{7}{3}|B_3^2| + |O_1'| + \tfrac{4}{3}|O_1'| \\
&= \tfrac{7}{3}|B_2| + \tfrac{7}{3}|B_3^1| + \tfrac{7}{3}|B_3^2| + \tfrac{7}{3}|O_1'| \\
&= \tfrac{7}{3}|OPT|,
\end{aligned}$$

where $|OPT|$ denotes the number of letters in OPT, and thus $|OPT| = |B_2| + |B_3^1| + |B_3^2| + |O_1'|$. That is, the number of isolates deleted by algorithm Approx-CMSR is at most $\frac{7}{3}$ times the optimum. Therefore, Approx-CMSR is a $\frac{7}{3}$-approximation algorithm. □

4 Conclusions

In this paper, we presented a $\frac{7}{3}$-approximation algorithm Approx-CMSR for the CMSR problem. The key design technique is a prioritized local greedy scheme to retain a number of isolates while deleting one or two isolates. Both the sequential order of the nine types of deleting-retaining operations, and their three levels of priorities, are crucial in the performance analysis. The performance analysis is done using a sequential amortized analysis, that attributes the isolates deleted by Approx-CMSR to the letters deleted in an optimal solution. Besides the proven performance ratio, this sequential amortized analysis technique is seemingly interesting itself.

The following instance shows that the performance ratio $\frac{7}{3}$ is tight. In this instance,

$$\begin{aligned}
G_1 &= \langle a, b, c, x_1, x_2, x_3, d, e, f, g, y_1, y_2, y_3, h, i, j, k, z_1, z_2, z_3, \ell \rangle, \text{ and} \\
G_2 &= \langle k, \ell, i, x_1, y_1, z_1, j, c, d, a, x_2, y_2, z_2, b, g, h, e, x_3, y_3, z_3, f \rangle.
\end{aligned}$$

G_1 and G_2 have no type-0 common substrings, neither deleting one or two letters from them leads to a novel type-1 common substring. Therefore, Approx-CMSR deletes all the letters. On the other hand, deleting 9 letters, $x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3$, leads to

$$\begin{aligned}
G_1^* &= \langle a, b, c, d, e, f, g, h, i, j, k, \ell \rangle, \text{ and} \\
G_2^* &= \langle k, \ell, i, j, c, d, a, b, g, h, e, f \rangle,
\end{aligned}$$

which can be partitioned into six common length-2 substrings: $a \bullet b$, $c \bullet d$, $e \bullet f$, $g \bullet h$, $i \bullet j$, and $k \bullet \ell$. In fact, one can verify that this is an optimal solution to the instance. It follows that the performance ratio of Approx-CMSR on this instance is $\frac{21}{9} = \frac{7}{3}$. Based on this 21-letter instance, one can construct an infinite series of instances on which the performance ratio of Approx-CMSR is $\frac{7}{3}$.

Acknowledgment

This research is partially supported by NSERC. The authors would like to thank Dr. Binhai Zhu for insightful discussion.

References

[1] R. Bar-Yehuda, M. M. Halldórsson, J. S. Naor, H. Shachnai, and I. Shapira. Scheduling split intervals. *SIAM Journal on Computing*, 36:1–15, 2006.

[2] L. Bulteau, G. Fertin, and I. Rusu. Maximal strip recovery problem with gaps: hardness and approximation algorithms. In *Proceedings of the 20th Annual International Symposium on Algorithms and Computation (ISAAC'09)*, LNCS 5878, pages 710–719, 2009.

[3] Z. Chen, B. Fu, M. Jiang, and B. Zhu. On recovering synthetic blocks from comparative maps. *Journal of Combinatorial Optimization*, 18:307–318, 2009.

[4] V. Choi, C. Zheng, Q. Zhu, and D. Sankoff. Algorithms for the extraction of syntheny blocks from comparative maps. In *Proceedings of the 7th International Workshop on Algorithms in Bioinformatics (WABI'07)*, pages 277–288, 2007.

[5] H. Jiang, Z. Li, G. Lin, L. Wang, and B. Zhu. Exact and approximation algorithms for the complementary maximal strip recovery problem. *Journal of Combinatorial Optimization*, 2010. Accepted for publication on November 3, 2010.

[6] M. Jiang. Inapproximability of maximal strip recovery. In *Proceedings of the 20th Annual International Symposium on Algorithms and Computation (ISAAC'09)*, LNCS 5878, pages 616–625, 2009.

[7] M. Jiang. Inapproximability of maximal strip recovery, ii. In *Proceedings of the 4th Annual Frontiers of Algorithmics Workshop (FAW'10)*, LNCS 6213, pages 53–64, 2010.

[8] L. Wang and B. Zhu. On the tractability of maximal strip recovery. *Journal of Computational Biology*, 17:907–914, 2010. (A one-page correction is to appear in January 2011).

[9] C. Zheng, Q. Zhu, and D. Sankoff. Removing noise and ambiguities from comparative maps in rearrangement analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4:515–522, 2007.