Tractability and approximability of maximal strip recovery*

Laurent Bulteau[1] Guillaume Fertin[1] Minghui Jiang[2†] Irena Rusu[1]

[1]Laboratoire d'Informatique de Nantes-Atlantique (LINA), UMR CNRS 6241
Université de Nantes, 2 rue de la Houssinière, 44322 Nantes Cedex 3, France
{Laurent.Bulteau,Guillaume.Fertin,Irena.Rusu}@univ-nantes.fr

[2]Department of Computer Science, Utah State University, Logan, UT 84322, USA
mjiang@cc.usu.edu

April 2, 2012

Abstract

An essential task in comparative genomics is to decompose two or more genomes into synteny blocks that are segments of chromosomes with similar contents. Given a set of d genomic maps each containing the same n markers without duplicates, the problem MAXIMAL STRIP RECOVERY (MSR) aims at finding a decomposition of the genomic maps into synteny blocks (strips) of the maximum total length ℓ, by deleting the minimum number $k = n - \ell$ of markers which are likely noise and ambiguities. In this paper, we present a collection of new or improved FPT and approximation algorithms for MSR and its variants. Our main results include a $2^{O(d\delta\ell)}\text{poly}(nd)$ time FPT algorithm for -gap-MSR-d, a $2.36^k\text{poly}(nd)$ time FPT algorithm for both CMSR-d and -gap-CMSR-d, and a $(d + 1.5)$-approximation algorithm for both CMSR-d and -gap-CMSR-d.

Keywords: Bioinformatics; Comparative genomics; Synteny blocks; Approximation algorithms; Parameterized complexity.

1 Introduction

An essential task in comparative genomics is to decompose two or more genomes into synteny blocks that are segments of chromosomes with similar contents. Synteny blocks represent units of the genomes that have not been disrupted by large-scale rearrangements such as reversals and transpositions, and thus form the input for genome rearrangement algorithms. They also give useful clues regarding the role of each gene, since genes belonging to the same synteny block often produce proteins with related functions. Extracting synteny blocks from genomic maps, however, is a non-trivial task when the genomic maps contain noise and ambiguities, which need to be removed before we can give a precise synteny block decomposition. This motivates the problem MAXIMAL STRIP RECOVERY (MSR) [9]: to delete a set of markers (genes) from the genomic maps until the remaining markers can be partitioned into a set of strips (synteny blocks) of maximum total length.

We review some definitions. A genome consists of one or more chromosomes; each chromosome is a sequence of genes. Correspondingly, a *genomic map* consists of one or more sequences of gene markers. Each marker is a signed integer representing a gene: the absolute value of the integer represents the family

*A preliminary version of this article appeared in CPM 2011.

†Corresponding author.

of the gene; the sign of the integer represents the orientation. A marker has *duplicates* if it is contained more than once in some genomic map, possibly in different orientations. A *strip* of $d \geq 2$ genomic maps is a sequence of at least two markers appearing consecutively in each map, such that the order of the markers and the orientation of each marker are either both preserved or both reversed. The *reversed opposite* of a sequence $s = \langle x_1, \ldots, x_h \rangle$ is $-s = \langle -x_h, \ldots, -x_1 \rangle$. The problem MSR on d input maps is the following maximization problem MSR-d [2, 9]:

PROBLEM MSR-d
INPUT: d genomic maps $G_1, \ldots, G_d$ each containing the same n markers without duplicates.
SOLUTION: d subsequences $G'_1, \ldots, G'_d$ of $G_1, \ldots, G_d$ respectively, each containing the same ℓ markers, such that all the markers in $G'_1, \ldots, G'_d$ can be partitioned into strips.
PARAMETER: the number ℓ of selected markers.

The maximization problem MSR-d that maximizes the parameter ℓ, the number of selected markers, has a *complement* minimization problem called CMSR-d [8, 6] that minimizes the parameter $k = n - \ell$, the number of deleted markers. For genomic maps of close species with few errors, k can be much smaller than ℓ, thus approximation and FPT algorithms are sometimes more relevant for CMSR than for MSR.

Given d subsequences $G'_1, \ldots, G'_d$ of d genomic maps $G_1, \ldots, G_d$, respectively, the *gap* between two consecutive markers a and b of G'_i is the number of markers appearing between a and b in G_i, a and b excluded. The *gap* of a strip s is the maximum gap between any two consecutive markers of s in any map G'_i. The deleted markers between markers of a strip correspond to noise and ambiguities, which occur infrequently. A synteny block is a segment of chromosomes that remain undisrupted by genome rearrangements during evolution. Consecutive elements of a synteny block can only be separated in a data set due to noise and ambiguities. Thus a strip having a large gap is unlikely to correspond to a synteny block. This leads to the following gap-constrained variant of MSR-d:

PROBLEM -gap-MSR-d
INPUT: d genomic maps $G_1, \ldots, G_d$ each containing the same n markers without duplicates.
SOLUTION: d subsequences $G'_1, \ldots, G'_d$ of $G_1, \ldots, G_d$ respectively, each containing the same ℓ markers, such that the markers in $G'_1, \ldots, G'_d$ can be partitioned into strips, and such that each strip has gap at most .
PARAMETER: the number ℓ of selected markers.

No doubt that MSR-d is a more elegant problem from a theoretical perspective, but -gap-MSR-d could be more relevant in biological applications. The gap-constrained variant of CMSR-d, denoted -gap-CMSR-d, can be similarly defined. Similarly to MSR-d and CMSR-d, the parameter for -gap-MSR-d is ℓ, and the parameter for -gap-CMSR-d is k. In most cases, and d are assumed to be constants, although our FPT algorithm in Theorem 3 does not depend on this assumption and can take and d as parameters besides ℓ. There is no known direct reduction from -gap-MSR-d to MSR-d or vice versa. Although the gap constraint appears to be an additional burden that the algorithm has to take care of, it also limits the set of candidate strips and their intersection pattern, especially when is small, which may make the problem easier to handle.

The following is an example of the problem MSR on three genomic maps G_1, G_2, G_3 and an optimal solution of three subsequences G'_1, G'_2, G'_3:

$$
\begin{aligned}
G_1 &= \quad 1 \quad 5 \quad -3 \quad 2 \quad 6 \quad 4 \quad 8 \quad 7 \\
G_2 &= \quad 1 \quad 5 \quad -3 \quad -8 \quad 7 \quad -6 \quad 2 \quad 4 \\
G_3 &= -8 \quad 2 \quad 7 \quad -6 \quad -4 \quad 3 \quad -5 \quad -1
\end{aligned}
$$

$$
\begin{aligned}
G'_1 &= \quad 1 \quad 5 \quad -3 \quad 6 \quad 8 \\
G'_2 &= \quad 1 \quad 5 \quad -3 \quad -8 \quad -6 \\
G'_3 &= -8 \quad -6 \quad 3 \quad -5 \quad -1
\end{aligned}
$$

Problem		Best FPT algorithm (running time)	
-gap-MSR-d		$O(2^t td^{\ 2} + nd^{\ })$ with $t = `(1 + \frac{3}{2}d^{\ })$	[Theorem 3.1, Section 3]
CMSR-d		$2{:}36^k \text{poly}(nd)$	[Theorem 4.1, Section 4]
-gap-CMSR-d	(≥ 2)	$2{:}36^k \text{poly}(nd)$	[Theorem 4.1, Section 4]
1-gap-CMSR-d		$2^k \text{poly}(nd)$	[Theorem 5.1, Section 5]

Problem		Best approximation ratio	
MSR-d		$2d$	[2, 6]
-gap-MSR-d	(≥ 4)	$2d$	[1]
1-gap-MSR-d	($d \geq 3$)	$0{:}75d + 0{:}75 +$	[Theorem 8.1, Section 8]
1-gap-MSR-2		$1{:}8$	[1]
2-gap-MSR-d		$1{:}5d +$	[Theorem 8.1, Section 8]
3-gap-MSR-d		$1{:}5d + 0{:}75 +$	[Theorem 8.1, Section 8]
CMSR-d	($d \geq 3$)	$d + 1{:}5$	[Theorem 6.1, Section 6]
CMSR-2		$7{=}3$	[7]
-gap-CMSR-d		$d + 1{:}5$	[Theorem 6.1, Section 6]
1-gap-CMSR-2		$2{:}778$	[Theorem 7.2, Section 7]

Table 1: Positive results for variants of MSR.

Here the markers $2; 4; 7$ are deleted, and the markers $1; 3; 5; 6; 8$ are selected in two strips $\langle 1; 5; -3\rangle$ and $\langle 6; 8\rangle$ of G'_1, G'_2, G'_3. The gap of the strip $\langle 1; 5; -3\rangle$ is 0. The gap of the strip $\langle 6; 8\rangle$ is 2, since there are 2 markers between -8 and -6 in G_3.

For the four variants of the maximal strip recovery problem, MSR-d, CMSR-d, -gap-MSR-d, and -gap-CMSR-d, several hardness results have been obtained [2, 8, 6, 1, 5], and a variety of algorithms have been developed, including heuristics [9], approximation algorithms [2, 1, 10, 4, 7], and FPT algorithms [8, 10, 4]. The previous hardness results regarding these problems are summarized in the following:

- [6, 1]: MSR-d, CMSR-d, -gap-MSR-d, and -gap-CMSR-d are APX-hard for any $d \geq 2$ and ≥ 2, even if all markers appear in positive orientation in all genomic maps; 1-gap-MSR-d and 1-gap-CMSR-d are NP-hard for any $d \geq 2$.
- [5]: MSR-d is W[1]-hard for any $d \geq 4$, even if all markers appear in positive orientation in all genomic maps.

On the positive side, it was known that MSR-d admits a $2d$-approximation algorithm for any $d \geq 2$ [2, 6], and that -gap-MSR-d admits a $2d$-approximation algorithm for any $d \geq 2$ and ≥ 1 and a $1{:}8$-approximation algorithm for $d = 2$ and $= 1$ [1]. Also, along some very recent development [10, 4, 7] on the CMSR problem parallel to our work, Lin et al. [7] presented a $7{=}3$-approximation algorithm for CMSR-2, which is based on an interesting idea called local amortization with re-weighting.

In this paper, we present a panel of new or improved FPT and approximation algorithms for the many variants of the maximal strip recovery problem. The current best results, including our contribution, are summarized in Table 1.

2 Preliminaries

We assume without loss of generality that all markers in the first input map G_1 have positive sign. Unless explicitly noted, our uses of some standard terms such as *solution* and *optimal solution*, our previous

definitions of *strip* and *gap*, as well as other definitions that we will introduce in this section — all these apply to some "current" set of genomic maps $G'_1; :: : ; G'_d$ implicit from context, which can be either the set of original maps $G_1; :: : ; G_d$ given as input, or some set of reduced maps (subsequences obtained from $G_1; :: : ; G_d$ by deleting some markers) during or after the execution of a recursive algorithm.

If a maximal sequence of markers form a strip in some maps $G'_1; :: : ; G'_d$, then these markers are either all selected or all deleted in *any* optimal solution for these maps. This is because any solution that includes only a subset of the markers in a strip can be extended to a better solution to include all markers in that strip. Hence these markers can be treated as an atomic unit, and called a *super-marker*, whose *size* is the number of markers it contains. Note that the size of a super-marker is always at least 2. A marker that does not belong to any super-marker is a *single-marker*. We use the term *single-super-marker* to refer to either a single-marker or a super-marker. A common (sometimes implicit) step of our algorithms is to partition the markers in the current maps into single-super-markers. If the current maps contain only super-markers, then we have a straightforward decomposition into strips, without deleting any marker.

Let x and y be two single-super-markers of some maps $G'_1; :: : ; G'_d$. We write $S^i(x; y)$ for the set of markers appearing between x and y in map G'_i. We say that y *follows* x in map G'_i if one of $\langle +x; +y\rangle$, $\langle -y; -x\rangle$ is a subsequence of G'_i. For -gap-CMSR-d, we add the constraint that the number of markers appearing between x and y in the original maps is at most (x and y excluded). For multichromosomal genomes, that is, when a genomic map consists of several sequences of gene markers (several chromosomes), we require that x and y belong to the same chromosome in G'_i. We define the relation "*y precedes x*" symmetrically.

We say that y is a *candidate successor* of x, and we write $x \prec y$, if y follows x in all maps, and if there is no other y' such that for all $1 \leq i \leq d$, y' follows x in G'_i and appears in $S^i(x; y)$. Note that if $\langle x; y\rangle$ is part of some strip in an optimal solution, then y is a candidate successor of x. We define symmetrically the candidate predecessors. Note that y is a candidate successor of x if and only if x is a candidate predecessor of y. If y is a candidate successor or predecessor of x, gap$(x; y)$ is the set of all markers appearing in $S^i(x; y)$ for some $1 \leq i \leq d$. In the example given in Section 1, we have $6 \prec 8$, and gap$(6; 8) =$ gap$(8; 6) = \{2; 4; 7\}$.

The following lemma gives some basic properties of the function gap:

Lemma 2.1. (a) *Let $u; v; w$ be three markers or single-super-markers. If u and v are two candidate successors of w with $u \neq v$, then $u \in$* gap$(w; v)$ *and $v \in$* gap$(w; u)$. (b) *Let u and v be two single-super-markers. If $u \prec v$ or $u \succ v$, then* gap$(u; v)$ *is not empty.*

Proof. (a) By definition of a candidate successor, there exists some i such that $v \in S^i(w; u)$, and some j such that $u \in S^j(w; v)$. Assume without loss of generality that w has positive sign in G_i. Then there exist sequences $s_{w,u}$ and $s_{w,v}$, using markers of $S^i(w; u)$ and $S^i(w; v)$ respectively, such that both $w\ s_{w,u}\ u$ and $w\ s_{w,v}\ v$ appear in map G_i. We now compare $|S^i(w; u)|$ and $|S^i(w; v)|$:

- if $|S^i(w; u)| = |S^i(w; v)|$, then $s_{w,u} = s_{w,v}$ and $u = v$ (this case is impossible),
- if $|S^i(w; u)| < |S^i(w; v)|$, then $s_{w,u}\ u$ is a prefix of $s_{w,v}$, and $u \in S^i(w; v) \subseteq$ gap$(w; v)$,
- if $|S^i(w; u)| > |S^i(w; v)|$, then $s_{w,v}\ v$ is a prefix of $s_{w,u}$ and $v \in S^i(w; u)$ (this case is impossible).

Likewise, using map G_j, we have $v \in S^j(w; u) \subseteq$ gap$(w; u)$. This proves the first property.

(b) Assume without loss of generality that $u \prec v$ (the other case $u \succ v$ is symmetric). If gap$(u; v) = \emptyset$, then for all i, $S^i(u; v) = \emptyset$, and hence either $\langle +u; +v\rangle$ or $\langle -v; -u\rangle$ appears in map G_i. It follows that $\langle u; v\rangle$ could form a super-marker: a contradiction. □

3 FPT algorithm for δ-gap-MSR-d

In this section, we present the first FPT algorithm for -gap-MSR-d with the parameter ℓ. Recall that without the gap constraint, MSR-d with the parameter ℓ is W[1]-hard for any $d \geq 4$. In sharp contrast to the W[1]-hardness of MSR-d, we obtain a somewhat surprising result that -gap-MSR-d is in FPT, where ℓ is the parameter, and and d are constants. In fact, our FPT algorithm for -gap-MSR-d works even if d and are not constants: -gap-MSR-d is in FPT even with three combined parameters d, and ℓ.

Theorem 3.1. *Algorithm 1 finds an optimal solution for* -gap-MSR-d *for any* $d \geq 2$ *and* ≥ 1, *in time* $O(2^t t d^{\ 2} + n d^{\ })$, *where* $t = \ell(1 + \frac{3}{2} d^{\ })$.

Algorithm 1 FPT algorithm for -gap-MSR-d

1: Gather all pairs of markers (u, v) such that $u \prec v$. Such pairs are called *candidate pairs*.
2: For each marker u, create a boolean variable x_u.
3: For each candidate pair (u, v), create a conjunctive boolean formula $f_{u,v} = x_u \wedge x_v \wedge \neg x_{g_1} \wedge \cdots \wedge \neg x_{g_s}$, where $g_1, \ldots, g_s$ are the markers in gap(u, v).
4: Delete the variables that do not appear in any formula or appear only in negative form in the formulas.
5: Enumerate all possible assignments to the remaining variables to find an optimal assignment that maximizes the number of variables appearing in positive form in at least one satisfied formula. Delete all markers whose variables are not assigned true values.
6: Return the resulting genomic maps.

Our algorithm is based on a simple idea: create a boolean variable for each marker (where true means the marker is selected in a solution, false that it is unselected), then test all possible assignments to find an optimal solution. To reduce the time complexity of this brute-force approach, we add a pruning step (line 4) to delete certain variables whose markers cannot appear in any optimal solution. The remaining variables form a "core" on which we can find an optimal solution in FPT time.

The correctness of the algorithm is deduced from the fact that each marker selected in a solution corresponds to a variable appearing in positive form in at least one formula, thus all optimal solutions are kept during the pruning step (line 4), and are discovered during the exhaustive enumeration (line 5).

Given an optimal solution, which selects ℓ markers, we call a marker *active* if it appears within distance at most from a selected marker in some map. Then each map contains at most $\ell + \frac{\ell}{2}$ unselected active markers: at most after each selected marker, and at most before the first marker of each strip (note that the number of strips of this optimal solution is at most ℓ=2). The total number of active markers is at most $\ell + d(\ell + \frac{\ell}{2}) = \ell(1 + \frac{3}{2} d^{\ })$.

The pruning step in line 4 depends on the crucial observation that a non-active marker can never appear in positive form. Suppose for contradiction that a non-active marker u appears in a candidate pair with some marker v. Then u is at distance at most $+1$ from v in each map. Since u, as a non-active marker, must be at distance at least $+1$ from the selected markers in all maps, no selected markers can appear between u and v in any map, thus we can extend the optimal solution by selecting both u and v, a contradiction.

Note that in line 4 the variables appearing at least once in positive form are never deleted, hence no formula becomes empty after deleting the variables that appear only in negative form. After line 4, the number of remaining variables is at most the number of active markers, which is at most $t = \ell(1 + \frac{3}{2} d^{\ })$. Correspondingly, the number of formulas is at most $t(\ +1)$, because any candidate pair consists of an active marker and one of the $+1$ markers immediately following it in the first map. Each formula contains at most $d\ +2$ variables.

The time complexity of line 1 is $O(n d^{\ })$. In lines 2 and 3, the variables can be created in time $O(n)$, and the formulas can be created in time $O(t(\ +1)(d\ +2)) = O(t d^{\ 2})$. Similarly, line 4 can be executed

in time $O(n + td^{\ 2})$. Finally, line 5 can be executed in time $O(2^t t(\ \ +1)(d\ \ +2)) = O(2^t td^{\ 2})$, so the overall time complexity is $O(2^t td^{\ 2} + nd\ \ )$.

4 FPT algorithm for CMSR-d and δ-gap-CMSR-d

In this section, we design an FPT algorithm for CMSR-*d* and -gap-CMSR-*d*, where the parameter is *k*, the number of deleted markers in the optimal solution.

Since super-markers are already strips in the input genomic maps, one may naturally be tempted to come up with the following algorithm. First, find all super-markers, and add them to the solution. Then, delete a subset of single-markers until all markers in the resulting maps can be partitioned into strips. The correctness of this algorithm for finding an exact solution, however, depends on the assumption that in some optimal solution no super-marker needs to be deleted, which is false as can be seen in the following counter-example:

$$\begin{array}{lccccccc} G_1 = & 4 & 1 & 2 & 3 & 5 & 6 & 7 \\ G_2 = & 6 & -3 & -2 & -1 & 7 & 4 & 5 \end{array}$$

Here $\langle 1; 2; 3\rangle$ forms a super-marker, but the optimal solution deletes $\langle 1; 2; 3\rangle$ and selects $\langle 4; 5\rangle$ and $\langle 6; 7\rangle$ instead. An easy generalization of this counter-example shows that any super-marker of size strictly less than $2d$ is not guaranteed to be always selected in some optimal solution. Note that on the other hand, longer super-markers, of size at least $2d$, are always selected in some optimal solution, see e.g. [4, Lemma 1].

We observe that an FPT algorithm for CMSR-*d* and -gap-CMSR-*d* can be easily obtained using the bounded search tree method. In any feasible solution for the two problems, a single-marker *x* must be either deleted or selected. If *x* is selected, then at least one of its neighbors must be deleted. Since *x* has at most $2d$ neighbors (at most two in each map), this leads to a very simple algorithm running in time $(2d+1)^k \text{poly}(nd)$. Parallel to our work, Jiang et al. [4] presented an FPT algorithm running in time $3^k \text{poly}(nd)$. We next describe a carefully tuned FPT algorithm running in time $2.36^k \text{poly}(nd)$. For convenience, we consider the decision problem associated with CMSR-*d* and -gap-CMSR-*d*, for which the parameter *k* is part of the input.

Theorem 4.1. *Algorithm 2 finds an exact solution for the decision problems associated with* CMSR-*d and* -gap-CMSR-*d, for any* ≥ 1 *and* $d \geq 2$*, in time* $c^k \text{poly}(nd)$*, where* $c < 2.36$ *is the unique real root of the equation* $2c^{-1} + 2c^{-3} = 1$*.*

It is interesting to note that although the two problems MSR-*d* and -gap-MSR-*d* have very different complexities when parameterized by ℓ, their complements CMSR-*d* and -gap-CMSR-*d* are both tractable when parameterized by *k*.

We describe the intuition behind Algorithm 2. As already noted, we will explore a bounded search tree as follows: in each node we consider a single-marker *x*, and we explore the branches corresponding to the cases where *x* is deleted and where it is selected in a strip with each possible candidate successor or predecessor. This search tree has bounded depth (in each branch we delete at least one marker, and we stop after deleting *k* markers) and degree (each single-marker has at most $2d$ candidate successors or predecessors). In order to improve the complexity of this algorithm, we aim at (1) choosing *x* so that we may delete a maximum number of markers in the subsequent recursive calls (thus reducing the depth of the subtree), and (2) pointing out special cases where we may ignore some branches of the search tree without loosing an optimal solution (thus reducing the degree). For objective (1) we choose *x* as the first single-marker in the first map, hence, the gap between *x* and a candidate predecessor consists mostly of super-markers, thus increasing the number of markers to be deleted in the corresponding branches. For objective (2), we provide a number of technical lemmas (Lemma 4.2 to Lemma 4.5) which allow us to reduce the degree of some "worst-case" nodes. In some situations, we find a marker which is necessarily deleted. In others, we identify a good candidate

predecessor or successor, which we may select to generate a solution at least as good as with any other candidate.

Algorithm 2 FPT algorithm for -gap-CMSR-d and CMSR-d

Input: d genomic maps $G_1; : : : ; G_d$ each containing the same n markers without duplicates, and two parameters $k \in \mathbb{N}$, $\in \mathbb{N} \cup \{\infty\}$

1: **return** recurse($G_1; : : : ; G_d; k;$; false)

Function recurse($G_1; : : : ; G_d; k;$; skip_step_2b): boolean

1: **if** $k < 0$ **then**
2: **return** false
3: Partition the markers into single-super-markers.
4: **if** there exists at least one single-marker in G_1 **then**
5: $x \leftarrow$ the left-most single-marker in G_1
6: **else**
7: **return** true
8: $s \leftarrow$ the first single-super-marker following x in G_1
9: *// 1: Assume x is deleted in the optimal solution*
10: Create $G'_1; : : : ; G'_d$ by removing x from $G_1; : : : ; G_d$.
11: **if** recurse($G'_1; : : : ; G'_d; k - 1;$; false) **then**
12: **return** true
13: *// 2: Assume x is part of a strip in the optimal solution*
14: $Y \leftarrow \{$ single-super-marker $y \mid x \prec y\}$ *// the set of candidate successors*
15: $Z \leftarrow \{$ super-marker $z \mid z \prec x\}$ *// the set of candidate predecessors*
16: **if** $\exists w_0 \in Y \cup Z$ a super-marker s.t. $(x; w_0)$ satisfies the conditions of Lemma 4.2 **then**
17: Create $G'_1; : : : ; G'_d$ by removing the marker in gap$(x; w_0)$ from $G_1; : : : ; G_d$.
18: **return** recurse($G'_1; : : : ; G'_d; k - 1;$; false)
19: **if** $\exists s_0$ a single-marker s.t. $(x; s_0)$ satisfies the conditions of Lemma 4.3 **then**
20: Create $G'_1; : : : ; G'_d$ by removing s_0 from $G_1; : : : ; G_d$.
21: **return** recurse($G'_1; : : : ; G'_d; k - 1;$; false)
22: *// 2.a: Assume x is not at the end of its strip*
23: **if** $Y \neq \emptyset$ **then**
24: **if** recurse_2a($Y; x; G_1; : : : ; G_d; k;$) **then**
25: **return** true
26: *// 2.b: Assume x is at the end of its strip*
27: **if** $Z \neq \emptyset$ **and** skip_step_2b=false **then**
28: **if** recurse_2b($Z; x; s; G_1; : : : ; G_d; k;$) **then**
29: **return** true
30: **return** false

4.1 Some technical lemmas

The efficiency of Algorithm 2 is made possible by several optimizations justified by the following four lemmas. These lemmas are all based on very simple observations. Note that although we consider the decision problem for simplicity, Algorithm 2 can be adapted to directly return the actual solution, instead of "true", when the input instance indeed has a solution of size k. Recall that the relation $\prec$ in lines 14–15 is defined for markers in the original maps — it remains unchanged through recursive calls, and can be

Algorithm 2 (continued)

Function recurse_2a($Y; x; G_1; : : : ; G_d; k;$): boolean

if $\exists y_0 \in Y$ s.t. y_0 satisfies the conditions of Lemma 4.4 **then**
  if $\in \mathbb{N}$ **and** y_0 is a single-marker **then**
    Replace y_0 by the unspecified marker $[y_0 \mid Y\,]$.
  $Y_0 \leftarrow \{y_0\}$
else
  $Y_0 \leftarrow Y$
for all $y \in Y_0$ **do**
  Create $G'_1; : : : ; G'_d$ by removing all markers in gap$(x; y)$ from $G_1; : : : ; G_d$.
  if recurse($G'_1; : : : ; G'_d; k - |\text{gap}(x; y)|;$; false) **then**
    return true
return false

Function recurse_2b($Z; x; s; G_1; : : : ; G_d; k;$): boolean

if $\exists z_0 \in Z$ s.t. z_0 satisfies the conditions of Lemma 4.5 **then**
  $Z_0 \leftarrow \{z_0\}$
else
  $Z_0 \leftarrow Z$
for all $z \in Z_0$ **do**
  if z ends with an unspecified marker $[y_0 \mid Y\,]$ **and** $\exists y_1 \in Y$ s.t. $y_1 \prec x$ **then**
    Replace the unspecified marker $[y_0 \mid Y\,]$ by y_1.
  Create $G'_1; : : : ; G'_d$ by removing all markers in gap$(x; z)$ from $G_1; : : : ; G_d$.
  skip_next_step_2b $\leftarrow$ s exists **and** s is a single-marker **and** $s \in$ gap$(x; z)$
  if recurse($G'_1; : : : ; G'_d; k - |\text{gap}(x; z)|;$; skip_next_step_2b) **then**
    return true
return false

precomputed.

Lemma 4.2. *Let x be a single-marker and w a super-marker. If x is selected in an optimal solution, and w is a candidate successor or predecessor of x with exactly one marker in* $\text{gap}(x; w)$*, then there is an optimal solution where the marker in* $\text{gap}(x; w)$ *is deleted.*

Proof. Assume that w is a candidate successor of x, the case where it is a candidate predecessor being symmetric.

Let v be the single-marker such that $\text{gap}(x; w) = \{v\}$, i.e., in some map G_i, one of $\langle +x; \pm v; +w\rangle$ or $\langle -w; \pm v; -x\rangle$ appears. In the case where no optimal solution selects both x and v, the lemma is obviously true since in any solution where x is selected, v must be deleted. It remains to consider the cases where an optimal solution O exists such that both x and v are selected. Since any strip of length $p \geq 4$ can be split into two shorter strips of lengths 2 and $(p-2)$, we can assume without loss of generality that all strips have lengths 2 or 3.

First case: x and v appear in the same strip. Then v is a candidate successor of x and w is not selected in O, since by Lemma 2.1a, $w \in \text{gap}(x; v)$. Create O' by removing this strip from O, the total size decreases by at most 3. Then no marker in $\{x; w\} \cup \text{gap}(x; w)$ is selected in O': we can add the strip xw to obtain a feasible solution of size greater than or equal to that of O, since w is a super-marker, where v is deleted.

Second case: x and v appear in different strips. Looking at map G_i, we see that v is at one end of its strip, and either (a) w is deleted or (b) w is in the same strip as v, and v precedes w. In case (a) we delete v (plus a second marker if v is in a length-2 strip), and add w at the end of the strip containing x: we again have an optimal solution where v is deleted. We now show that case (b) is absurd: since w is a candidate successor of both x and v, then x and v are candidate predecessors of w. However, by Lemma 2.1a, $x \in \text{gap}(w; v)$, so this contradicts the fact that x is selected and $w; v$ are in the same strip. □

Lemma 4.3. *Let x be a single-marker and s a single-super-marker. If s appears in* $\text{gap}(x; w)$ *for each w that is a candidate successor or predecessor of x, then s itself cannot be a candidate successor or predecessor of x, and any solution selecting x deletes s.*

Proof. We first prove that s is not a candidate successor or predecessor of x: otherwise, we would have $s \in \text{gap}(x; s)$, which is impossible.

Consider the strip containing x in any feasible solution. If x is at the end of this strip, it is preceded by a candidate predecessor z, $z \neq s$, and all markers in $\text{gap}(x; z)$, including s, are deleted. Otherwise x is followed in its strip by a candidate successor y, and again $s \in \text{gap}(x; y)$ is deleted. □

Lemma 4.4. *(In this lemma we assume there is no gap constraint.) Let x be a single-marker and y a candidate successor of x such that all markers in* $\text{gap}(x; y)$ *are single-markers and candidate successors of x. If x is part of some strip in an optimal solution, but not at the end of this strip, then there is an optimal solution where $\langle x; y\rangle$ is part of some strip.*

Proof. Let y_0 be the single-super-marker following x in the strip of the optimal solution, and y_1 be the successor of y_0 if it exists. If $y = y_0$, then the lemma is proved. Otherwise, $y_0 \in \text{gap}(x; y)$ (by Lemma 2.1a) and y_0 is a single-marker.

If y_1 does not exist, we can replace y_0 by y if we delete all markers in $\text{gap}(x; y) - \{y_0\}$. But since all these markers are candidate successors of x, they also appear in $\text{gap}(x; y_0)$ and are already deleted, hence the total size of the solution is unchanged.

Assume now that y_1 exists, we prove that y_1 is a candidate successor of y. First of all, $x; y; y_0; y_1$ appear in the same sequence of gene markers (in the same chromosome) in each map. Moreover, $x; y; y_1$ appear in this order in all maps: y and y_1 both appear after x, and y_1 cannot appear in any $S^i(x; y)$, otherwise $y_1 \in \text{gap}(x; y)$ and y_1 would be a candidate successor of x (which is absurd, since $y_0 \in S^j(x; y_1)$ for all

j). Since there is no gap constraint, y_1 is a candidate successor of y. We can replace y_0 by y if we delete all markers in $\Gamma = (\text{gap}(x;y) \cup \text{gap}(y;y_1)) - \{y_0\}$:

$$\begin{aligned}
\Gamma &= \left[\begin{array}{c} \\ i \end{array}\right. \left(S^i(x;y) \cup S^i(y;y_1) \right) - \{y_0\} \\
&= \left[\begin{array}{c} \\ i \end{array}\right. \left(S^i(x;y_1) \right) - \{y;y_0\} \\
&= \left[\begin{array}{c} \\ i \end{array}\right. \left(S^i(x;y_0) \cup S^i(y_0;y_1) \right) - \{y\} \\
&= (\text{gap}(x;y_0) \cup \text{gap}(y_0;y_1)) - \{y\}:
\end{aligned}$$

Then all markers in Γ are already deleted: we can replace y_0 by y without changing the solution size. □

Lemma 4.5. *Let x be the first single-marker in G'_1. Let z be a candidate predecessor of x such that all markers in* $\text{gap}(x;z)$ *are size-2 super-markers and candidate predecessors of x. If x appears at the end of a strip in an optimal solution, then there is an optimal solution where $\langle z;x \rangle$ is at the end of some strip.*

Proof. Let z_0 be the single-super-marker preceding x in the strip of the optimal solution, and z_1 be the one preceding z_0. If $z = z_0$, the lemma is proved. Otherwise, $z_0 \in \text{gap}(x;z)$, hence it is a size-2 super-marker.

If z_1 exists, then it is also a super-marker, since x is the first single-marker in G'_1, and we can split the strip between z_1 and z_0: hence we can assume that the strip containing x in the optimal solution is z_0x.

We can replace z_0 by z in z_0x by deleting all markers in $\text{gap}(x;z) - \{z_0\}$. Since all these markers appear in $\text{gap}(x;z_0)$, they are already deleted in the optimal solution. Moreover, $|z| \geq 2 = |z_0|$, so replacing z_0 by z does not reduce the solution size. □

In addition to these four optimizations, we also use a "delayed commitment" optimization which is the equivalent of Lemma 4.4 when we need to observe a gap constraint. We consider the case where x is part, but not at the end, of some strip in the optimal solution, and where y is a single-marker and a candidate successor of x such that all markers in $\text{gap}(x;y)$ are single-markers and candidate successors of x. In this case we delete all markers in $\text{gap}(x;y)$ to make $\langle x;y \rangle$ a strip, but keep the possibility of replacing y by any marker $y_1 \in \text{gap}(x;y)$, should necessity arise. We denote this unspecified marker by $[y \mid \text{gap}(x;y)]$.

4.2 Correctness of Algorithm 2

To prove the correctness of Algorithm 2, we also need the following lemma from [9]. We provide an easy proof for completeness.

Lemma 4.6. [9, Proposition 2] *We can decompose the strips of any optimal solution in such a way that (1) each strip contains at most 3 single-super-markers and (2) each strip containing 3 single-super-markers starts and ends with a single-marker.*

Proof. Let s be a strip containing h single-super-markers: $s = s_1 s_2 ::: s_h$. If $h \geq 4$, we can split s into two strips: $s_1 s_2$ and $s_3 s_4 ::: s_h$. We apply this until condition (1) is true. If $h = 3$ and s_1 (respectively s_3) is a super-marker, then we can split s into s_1 and $s_2 s_3$ (respectively $s_1 s_2$ and s_3). We can do this operation until condition (2) also becomes true. □

Let OPT be any optimal solution. Decompose the strips of OPT as in the above lemma. We show by induction that the solution found by Algorithm 2 has the same size as OPT. Let x be the left-most single-marker in G_1, then exactly one of the following three cases is true:

1: x is deleted in OPT,

2.a: There exists a single-super-marker y such that $\langle x; y\rangle$ is part of a strip in OPT,

2.b: There exists a super-marker z such that $\langle z; x\rangle$ is a strip in OPT.

Note that in case 2.b, z cannot be a single-marker since it is to the left of x in G_1. By our choice of x, case 2.a can be split into the following two subcases:

2.a.i: There exists a single-super-marker y such that $\langle x; y\rangle$ is a strip in OPT,

2.a.ii: There exists a single-super-marker y and a single-marker y' such that $\langle x; y; y'\rangle$ is a strip in OPT.

Refer to Algorithm 2. In case 1, a solution is found in lines 9–12 of the function recurse. In case 2, i.e. in the case where x is part of an optimal solution, if either Lemma 4.2 or Lemma 4.3 can be applied, then again a solution is found. Otherwise, we are in case 2.a or 2.b.

Suppose we are in case 2.a. If $y \in Y_0$, then the function recurse_2a tests a branch in which $\langle x; y\rangle$ becomes part of some strip. Otherwise, there exists some $y_0 \in Y$ satisfying the conditions of Lemma 4.4. If there is no gap constraint, y is replaced by y_0, which does not change the size of the solution. If there is a gap constraint, y is replaced by the unspecified marker $u = [y_0 \mid Y]$, and we look further in case 2.a.i or 2.a.ii.

In case 2.a.i, we can replace y by y_0 since $\text{gap}(x; y_0)$ has no more markers than $\text{gap}(x; y)$. In case 2.a.ii, we can replace y by any y_1 such that $x \prec y_1 \prec y'$, since $\text{gap}(x; y) \cup \{y\} \cup \text{gap}(y; y')$ is the same set as $\text{gap}(x; y_1) \cup \{y_1\} \cup \text{gap}(y_1; y')$. This is what happens in case 2.b of a subsequent recursive call in which y' becomes the left-most single-marker in G_1.

Suppose we are in case 2.b. If $z \in Z_0$, then the function recurse_2b tests a branch in which $\langle z; x\rangle$ becomes a strip. Otherwise, Lemma 4.5 can be applied, which leaves the size of the optimal solution unchanged. In line 9 of recurse_2b, if s becomes the left-most single-marker in G_1 in the next recursive call of recurse, it cannot be at the end of a strip because x is already at the end of a strip.

This completes the correctness proof.

4.3 An example on the behavior of the unspecified markers

We run Algorithm 2 on the following three maps, with the gap constraint $= 3$:

$$G_1 = \langle 1; 2; a; 4; 3; r; b\rangle$$

$$G_2 = \langle 1; 3 \quad 11\ 10\ 9091 \quad 5\ 454\ 0 \quad \mathrm{d}\ [(\ )] \quad 8\ 10\ 9091 \quad 4\ 849\ 0 \quad \mathrm{d}\ [($$

where r is a candidate successor of a. Hence the algorithm finds the solution consisting of the length-4 strip $\langle 1; 2; a; r\rangle$.

In another branch of the search tree, where a and r are deleted, we obtain the following maps:

$$G_1 = \langle (1; [2 \mid 2; 3; 4]); b\rangle$$
$$G_2 = \langle (1; [2 \mid 2; 3; 4]); b\rangle$$
$$G_3 = \langle (1; [2 \mid 2; 3; 4]); x; b\rangle$$

Here b is the left-most single-marker in G_1, and $(1\ [2 \mid 2; 3; 4])$ is a candidate predecessor of b with 3 and 4 (not with 2, since the gap between 2 and b in the original maps is $4 >$). Hence part (2.a) of Algorithm 2 deletes the markers in $\text{gap}(b; [2 \mid 2; 3; 4]) = \{x\}$, and replaces the unspecified marker $[2 \mid 2; 3; 4]$, e.g. by 3. Thus Algorithm 2 finds in another branch of the search tree the length-3 strip $\langle 1; 3; b\rangle$.

4.4 Complexity analysis of Algorithm 2

Let $T(k)$ be the complexity of the function recurse of Algorithm 2 with parameters k and skip_step_2b=false, and $T_{skip}(k)$ the complexity of this function with parameters k and skip_step_2b=true (the complexity here being the number of leaves in the search tree). The complexity of several parts of the algorithm depends on whether the single-super-marker s defined at line 8 is a single-marker: so we define a boolean variable s_single, which is true if s exists and is a single-marker, and false otherwise. We now compute the complexity of each part of the algorithm.

Part 1: The complexity from line 9 to 12 is $T(k-1)$.

Part 2 (lines 13 to 29): if one of the conditions from lines 16 and 19 is true, then the complexity here is $T(k-1)$. Otherwise, we need to analyze the complexity of parts 2.a (lines 22 to 25) and 2.b (lines 26 to 29).

Part 2.a: We write r for the number of single-super-markers in Y_0, and r' for the minimum size of $\text{gap}(x; y)$ for $y \in Y_0$, and y_0 the single-super-marker reaching this bound; then the complexity is at most $rT(k-r')$. We now bound r and r': first, by Lemma 2.1a, we already have $r' \geq r-1$. We now prove by contradiction that $r' > r-1$. Assume that $r' = r-1$, then the candidate successors of $Y - \{y_0\}$ are the only markers appearing in $\text{gap}(x; y_0)$, and they are all single-markers. Thus y_0 satisfies the conditions of Lemma 4.4, and $Y_0 = \{y_0\}$, $r = 1$ and $r' = 0$ (even if y_0 is replaced by an unspecified marker in the meantime). This is absurd, since $r' = |\text{gap}(x; y_0)|$ and $\text{gap}(x; y_0)$ is not empty by Lemma 2.1b. Thus $r' \geq r$ and the complexity of part 2.a is upper bounded by $r'T(k-r')$ with $r' \geq 1$.

Moreover, if s_single is false, then we show that we cannot have $r' = 1$. By contradiction again, suppose $r' = 1$. The super-marker s exists (otherwise no marker follows x in G_1, so $Y = \emptyset$), and it appears in $\text{gap}(x; y)$ for all $y \in Y - \{s\}$, then we necessarily have $y_0 = s$ and $|\text{gap}(x; y_0)| = 1$. This would mean that $(x; y_0)$ satisfies the conditions of Lemma 4.2, a contradiction.

Thus the complexity of part 2.a is at most $\max\{r'T(k-r') \mid r' \geq 1\}$ if s_single is true, and $\max\{r'T(k-r') \mid r' \geq 2\}$ otherwise.

Part 2.b: First note that all $z \in Z$ are super-markers. We denote by t the number of super-markers in Z, and by t' the minimum size of $\text{gap}(x; z)$ for $z \in Z_0$ (we write z_0 for the super-marker reaching this bound). By Lemma 2.1a, $\text{gap}(x; z_0)$ contains at least $t-1$ super-markers, thus $t' \geq 2(t-1)$. Moreover $t' \neq 0$ (Lemma 2.1b) and $t' \neq 1$ (otherwise $(x; z_0)$ would satisfy the conditions of Lemma 4.2); and one cannot have $t' = 2(t-1)$ for $t \geq 2$: if $t' = 2(t-1)$, then z_0 satisfies the conditions of Lemma 4.5, so $Z_0 = \{z_0\}$ and $t = 1$.

Hence the complexity of part 2.b is at most $\max\{T(k-2); \max\{tT(k-2t+1) \mid t \geq 2\}\}$. This is the best bound we obtain when s_single is false, but it can be improved when s_single is true.

Indeed, if s_single is true, we consider Z_1 the set of $z \in Z_0$ such that $s \in \text{gap}(x; z)$ and $Z_2 = Z_0 - Z_1$. We can see that Z_2 is not empty: otherwise $(x; s)$ would satisfy the conditions of Lemma 4.3. Several cases are possible:

- $t = 1$, then $Z_0 = Z_2$ contains only one super-marker z_0, and the complexity is $T_{skip}(k-2)$.
- $t \geq 2$: For each $z \in Z_1$, $\text{gap}(x; z)$ contains at least $(t-1)$ super-markers from $Z - \{z\}$ and the single-marker s, so the complexity is $T(k - 2(t-1) - 1)$. For $z \in Z_2$, it is $T_{skip}(k - 2(t-1) - 1)$.

Overall, the complexity of part 2.b in the case where s_single is true is at most:

$$\max\{T_{skip}(k-2); \max\{T_{skip}(k-t') + (t-1)\max\{T(k-t'); T_{skip}(k-t')\} \mid t \geq 2; t' = 2t-1\}\}$$

We can now show by induction that $T(k) \leq c^k$ and $T_{skip}(k) \leq c^k$, with $c \approx 2{:}3593$ (c is the real positive solution of $1 = 2c^{-1} + 2c^{-3}$) and $= 2/c \approx 0{:}8477$.

For part 2.a, we have the following upper bounds:

- for s_single = false
$$\max\{r'T(k-r') \mid r' \geq 2\} \leq 2c^{k-2}$$
- for s_single = true
$$\max\{r'T(k-r') \mid r' \geq 1\} \leq c^{k-1}$$

And for part 2.b:

- for s_single = false
$$\max\{T(k-2); \max\{tT(k-2t+1) \mid t \geq 2\}\} \leq c^{k-2}$$
- for s_single = true
$$\max\{T(k-t'); T_{skip}(k-t')\} \leq c^{k-t'} \text{ for all } t'$$
$$\max\{T_{skip}(k-t') + (t-1)c^{k-t'} \mid t \geq 2; t' = 2t-1\} \leq 2c^{k-4} + c^{k-3}$$
$$\max\{T_{skip}(k-2); 2c^{k-4} + c^{k-3}\} \leq 2c^{k-3}$$

We first look at the case where s_single = false:

$$T_{skip}(k)c^{-k} \leq c^{-1} + \max\{c^{-1}; 2c^{-2}\} = 2c^{-1} =$$
$$T(k)c^{-k} \leq c^{-1} + \max\{c^{-1}; 2c^{-2} + c^{-2}\} = 0{:}962\ldots < 1$$

Next for s_single = true:

$$T_{skip}(k)c^{-k} \leq c^{-1} + \max\{c^{-1}; c^{-1}\} = 2c^{-1} =$$
$$T(k)c^{-k} \leq c^{-1} + \max\{c^{-1}; c^{-1} + 2c^{-3}\} = 1$$

Thus we have $T(k) \leq c^k$ and $T_{skip}(k) \leq c^k$. This proves that the size of the search tree is bounded by $O(c^k)$, and each recursive call is done in polynomial time in n and d, so, altogether, the running time of Algorithm 2 is $c^k \text{poly}(dn)$.

As a final remark regarding Algorithm 2, an anonymous reviewer of an earlier version of this paper commented that perhaps some further properties of the optimal solution, besides those already described in our lemmas, might be used to improve the time complexity further. This may be true, but we believe that such improvement would require significantly different ideas.

5 FPT algorithm for 1-gap-CMSR-d

In this section we present an improvement of Algorithm 2 for the problem 1-gap-CMSR-d.

Theorem 5.1. *Algorithm 3 finds an exact solution for the decision problem associated with* 1-gap-CMSR-d*, for any* $d \geq 2$*, in time* $2^k \text{poly}(nd)$.

Note that, as for Algorithm 2, Algorithm 3 can be easily adapted to produce a full solution instead of simply returning "true", when the instance indeed has a solution of the right size.

Proof. Algorithm 3 is based on the following observation. Let x be the left-most single-marker in map 1, and assume it appears in an optimal solution, then there are two cases:

(1) x has a candidate predecessor z_0 (it is necessarily a super-marker). Then, by Lemma 5.2, we can delete all markers between x and z_0 in all maps, regardless of whether z_0 and x are in the same strip in the optimal solution. At least one such marker must exist.

(2) x has no candidate predecessor, then it must be in the same strip as a successor. With the gap constraint, x can have at most two successors. Using Lemma 5.3, we can choose one of them (y_0, in Algorithm 3).

This proves the correctness of the algorithm. Moreover, the complexity of the 1-gap-CMSR function with parameter k is at most $2^k \text{poly}(nd)$: it is polynomial except for at most two recursive calls, each with a parameter $k' < k$. Thus Theorem 5.1 is proved. □

Lemma 5.2. *(This lemma uses the gap constraint* $= 1$*.) Let* x *be a single-marker, and* z *a super-marker candidate predecessor of* x*. Then if an optimal solution selects* x*, it also deletes all markers in* $\text{gap}(x; z)$.

Proof. With the gap constraint, z is the only candidate predecessor of x, and it is selected in the optimal solution (like all super-markers).

Take $u \in \text{gap}(x; z)$, then u cannot be in the same strip as x (it is neither a candidate successor nor predecessor of x). Hence if u is selected in the optimal solution, then it is in the same strip as z and all markers of $\text{gap}(z; u)$, including x (see Lemma 2.1a), are deleted: a contradiction. So all markers in $\text{gap}(x; z)$ are deleted in the optimal solution. □

Lemma 5.3. *(This lemma uses the gap constraint* $= 1$*.) Let* x *be a single-marker with two candidate successors* a *and* b*. If* x *appears in an optimal solution, but not at the end of its strip, then there is an optimal solution where* $\langle x; c \rangle$ *is part of some strip, with* $c = \text{choose}(a; b)$ *(see Algorithm 3, parameters* $G_1; \ldots; G_d$ *are omitted).*

Proof. For simplicity, we assume without loss of generality that x has a positive sign in all maps. Otherwise, if x has a negative sign in some map G_i, we can replace this map by its reversed opposite.

If $= 1$ and x has two candidate successors a_1 and b_1, then in each map we have the sequence $\langle x; a_1; b_1 \rangle$ or $\langle x; b_1; a_1 \rangle$. Moreover, if both a_1 and b_1 have at least one candidate successor (respectively a_2 and b_2) with $a_2 \neq b_2$, then again only two patterns are possible in all maps: $\langle x; a_1; b_1; a_2; b_2 \rangle$ or $\langle x; b_1; a_1; b_2; a_2 \rangle$. We proceed with this construction recursively, until we reach a pair $(a_h; b_h)$ such that a_h and b_h do not have different candidate successors.

Assume that x is selected in a strip of an optimal solution, followed by h' markers in this strip, with $1 \leq h' \leq h$. Then these markers are either $\langle a_1; \ldots; a_{h'} \rangle$ or $\langle b_1; \ldots; b_{h'} \rangle$, and we can replace one sequence by the other without creating overlapping strips, so there are optimal solutions selecting $\langle x; c \rangle$ for $c = a_1$ and for $c = b_1$.

If $h' > h$, let u be the $h + 1^{\text{st}}$ marker following x in the strip (and assume without loss of generality that the first h selected markers are $\langle b_1; \ldots b_h \rangle$). Then u is a candidate successor of b_h, and either a_h has no

Algorithm 3 FPT algorithm for 1-gap-CMSR-d

Function 1-gap-CMSR($G_1; : : : ; G_d; k$): boolean

if $k < 0$ **then**
  return false
Partition the markers into single-super-markers.
if there exists at least one single-marker in G_1 **then**
  $x \leftarrow$ the left-most single-marker in G_1
else
  return true
// 1: Assume x is deleted in the optimal solution
Create $G'_1; : : : ; G'_d$ by removing x from $G_1; : : : ; G_d$.
if 1-gap-CMSR($G'_1; : : : ; G'_d, k - 1$) **then**
  return true
// 2: Assume x is selected in the optimal solution
if $\exists z_0 \prec x$ **then**
  Create $G'_1; : : : ; G'_d$ by removing all markers in gap$(x; z_0)$ from $G_1; : : : ; G_d$.
  return 1-gap-CMSR($G'_1; : : : ; G'_d; k - |\text{gap}(x; z_0)|$)
else if $\exists a \succ x$ **then**
  if $\exists b \succ x$ s.t. $b \neq a$ **then**
    $y_0 \leftarrow$ choose($G'_1; : : : ; G'_d; a; b$)
  else
    $y_0 \leftarrow a$
  Create $G'_1; : : : ; G'_d$ by removing all markers in gap$(x; y_0)$ from $G_1; : : : ; G_d$.
  return 1-gap-CMSR($G'_1; : : : ; G'_d; k - |\text{gap}(x; y_0)|$)
else
  return false

Function choose($G_1; : : : ; G_d; a; b$): single-marker

if $\exists a' \succ a$ **then**
  if $\exists b' \succ b$ **and** $b' \neq a'$ **then**
    if choose($G_1; : : : ; G_d; a'; b'$)$= a'$ **then**
      return a
    else
      return b
  else
    return a
else
  return b

Algorithm 4 $(d+1.5)$-approximation for -gap-CMSR-d and CMSR-d

1: $X \leftarrow \{$ triples of markers $(z;x;y) \mid z \prec y$ and gap$(z;y) = \{x\}\}$
2: Partition the markers into single-super-markers.
3: **for all** $(z;x;y) \in X$ **do**
4: **if** x, y and z are not deleted **and** y or z is a single-marker **then**
5: Delete x.
6: Re-create all super-markers.
7: Delete all remaining single-markers.
8: Return the resulting genomic maps.

candidate successor, or it has only u. In the first case, choose$(a_h;b_h) = b_h$, and choose$(a_1;b_1) = b_1$: this is the choice made in the optimal solution. In the second case, choose$(a_1;b_1) = a_1$, but in the strip of the optimal solution, we can replace $\langle x;b_1;\ldots;b_h;u\rangle$ by $\langle x;a_1;\ldots;a_h;u\rangle$ without creating incompatibilities, since we have gap$(a_h;u) = \{b_h\}$ and gap$(b_h;u) = \{a_h\}$. Thus there is also an optimal solution selecting $a_1;\ldots;a_h;u$ after x: this proves the lemma. □

6 Approximation algorithm for CMSR-d and δ-gap-CMSR-d

In this section, we present a $(d+1.5)$-approximation algorithm for the two minimization problems CMSR-d and -gap-CMSR-d. Recall that $2d$-approximation algorithms [2, 6, 1] were known for the two maximization problems MSR-d and -gap-MSR-d.

Theorem 6.1. *Algorithm 4 finds a $(d+1.5)$-approximation for* CMSR-d *and* -gap-CMSR-d *for any $d \geq 2$ and ≥ 1.*

Let k be the number of deleted markers in an optimal solution. Then the number of single-markers in the input maps is at most $(2d+1)k$ because each single-marker is either deleted or adjacent to a deleted marker. This immediately yields a $(2d+1)$-approximation algorithm: simply delete all single-markers. The following is a tight example for this algorithm (here the optimal solution deletes one single-marker x instead of all $2d+1$ single-markers):

$$\begin{array}{lccccc} G_1 = & z_d y_d & \cdots & z_3 y_3 & z_2 y_2 & z_1\, x\, y_1 \\ G_2 = & z_1 y_1 & z_2\, x\, y_2 & z_3 y_3 & \cdots & z_d y_d \\ G_3 = & z_1 y_1 & z_2 y_2 & z_3\, x\, y_3 & \cdots & z_d y_d \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ G_d = & z_1 y_1 & z_2 y_2 & z_3 y_3 & \cdots & z_d\, x\, y_d \end{array}$$

As we can see from the above example, after one single-marker is deleted, many other single-markers may be merged into strips. Algorithm 4 first identifies (line 1) all triples of markers $(z;x;y)$ such that z and y can be merged into a strip $\langle z;y\rangle$ after x is deleted. Such markers x are considered to be "cost-efficient" when z or/and y is a single-marker, since they allow, if deleted, to merge at least one single-marker into a super-marker. Thus the algorithm successively deletes (lines 2–6) those cost-efficient single-markers (each time reducing the total number of single-markers by at least 2), and finally removes (line 7) the remaining single-markers.

The approximation ratio analysis is non-trivial. We first give a number of definitions and easy remarks (inequalities (1) and (2)). We then bound on one hand the number of cost-efficient single-markers which are deleted by the algorithm but not by an optimal solution ($|D - O|$ in (3)), and on the other hand the number of single-markers which have been cost-efficient for the optimal solution but are not deleted as such

by the algorithm ($|R_1|$ in (4)). Finally, combining inequalities (1) to (4), we obtain a lower bound on the approximation ratio.

Lemma 6.2. *For each triple $(z; x; y)$ in the set X in Algorithm 4, at least one of the three markers $x; y; z$ must be deleted in any feasible solution.*

Proof. We prove the lemma by contradiction. Suppose that all three markers $x; y; z$ are selected in a solution. Assume without loss of generality that the sequence $\langle z; x; y\rangle$ appears in some map. Then x must be in the same strip as z or y. Assume without loss of generality that $\langle z; x\rangle$ is part of some strip. Then $z \prec x$. Recall that $z \prec y$. Thus x and y are both candidate successors of z. By Lemma 2.1a, we have $y \in \text{gap}(z; x)$, thus y must be deleted: a contradiction. □

We next prove the approximation ratio of Algorithm 4. Let O be the set of deleted markers in an optimal solution; $|O| = k$. For each marker $x \in O$, we define two sets $\Gamma_{succ}(x)$ and $\Gamma_{pred}(x)$ as follows. If x is followed by a marker y in a strip of O, $\Gamma_{succ}(x) = \text{gap}(x; y)$; otherwise x is the last marker of its strip, $\Gamma_{succ}(x) = \emptyset$. If x is preceded by a marker z in a strip of O, $\Gamma_{pred}(x) = \text{gap}(z; x)$; otherwise x is the first marker of its strip, $\Gamma_{pred}(x) = \emptyset$. Then, for each marker $x \in O$, define $(x) = |\Gamma_{succ}(x)| + |\Gamma_{pred}(x)|$, and for each marker $x \in O$, define $(x) = 0$.

Refer to Algorithm 4. Let D be the set of markers deleted in line 5, let S be the set of single-markers that are merged into super-markers in line 6, and let R be the set of markers deleted in line 7. Let $R_1 = \{r \in R \mid (r) = 1\}$ and $R_2 = \{r \in R \mid (r) \geq 2\}$. Note that if x is a single-marker at the beginning of the algorithm, then $(x) = 0$ if and only if $x \in O$. Thus we have a partition of R given by $R = (R \cap O) \cup R_1 \cup R_2$. Also note that each marker $x \in O$ is counted by at most twice in each map: at most once in some $\Gamma_{pred}(y)$, and at most once in some $\Gamma_{succ}(z)$. Thus we have the following inequality:

$$\sum_{x \text{ single-marker}} (x) \leq 2dk: \tag{1}$$

Each marker $x \in D$ has a corresponding triple $(z; x; y) \in X$, where z or y is a single-marker. After x is deleted in line 5, z and y are merged into the same super-marker in line 6. Thus we have the following inequality:

$$|D| \leq |S|: \tag{2}$$

For each marker $x \in D - O$, let (x) be an arbitrary marker in the non-empty set $\{z; x; y\} \cap O$; see Lemma 6.2. Obviously $(x) \neq x$, thus $(x) \in O - D$. We show that at most two markers in $D - O$ can have the same image by . Suppose that $(x_1) = (x_2) =$ for two different markers $x_1; x_2 \in D - O$, where x_1 is deleted before x_2 in Algorithm 4. Then the marker is merged into a super-marker after x_1 is deleted, and again merged into a larger super-marker after x_2 is deleted. Since a marker has at most two neighbors in a super-marker, is necessarily a single-marker before x_1 is deleted, so it belongs to S, indeed $S \cap O$. Moreover, after x_2 is deleted and is merged into a larger super-marker, cannot be adjacent to any other single-marker, say x_3. Therefore

$$|D - O| \leq |O - D| + |S \cap O|: \tag{3}$$

Let u be a marker such that $(u) = 1$. Then by definition of , u belongs to some strip in the optimal solution, and it has a neighbor $v =$ (u) in the same strip such that $\text{gap}(u; v)$ contains only one marker, say x. Note that $u; v \in O$ and $x \in O$. We claim that if u is a single-marker at the beginning of the algorithm, then either $u \in D \cup S$ or $v \in D$. This claim is clearly true if one of u or v is deleted by the algorithm in line 5. Otherwise, with $(v; x; u) \in X$ or $(u; x; v) \in X$, either x is not deleted because u is merged into a super-marker, or x is deleted: in both cases $u \in S$. This proves the claim. So for each $u \in R_1$, we have

$v \in D$, indeed $v \in D - O$. Note that there can be at most two markers u_1 and u_2 with the same image v by : the two neighbors of v in some strip in the optimal solution. Thus we have $|R_1| \leq 2|D - O|$. Moreover, if there are two markers u_1 and u_2 with the same image v, then $(v) \geq 2$. Therefore

$$|R_1| \leq \sum_{v \in D-O} (v): \tag{4}$$

Combining inequalities (1), (2), (3), and (4), the calculation in the following shows that the number of deleted markers, $|D| + |R|$, is at most $(d + 1.5)k$. Thus Algorithm 4 indeed finds a $(d + 1.5)$-approximation for -gap-CMSR-d and CMSR-d.

$$\begin{aligned}
2dk &\geq \sum_{x \text{ single-marker}} (x) \quad \text{by (1)} \\
&= \sum_{x \in D-O} (x) + \sum_{x \in S-O} (x) + \sum_{x \in R_1} (x) + \sum_{x \in R_2} (x) \\
&\geq \sum_{x \in D-O} (x) + |S - O| + |R_1| + 2|R_2| \\
&\geq |S - O| + 2|R_1| + 2|R_2| \quad \text{by (4)}
\end{aligned}$$

$$\begin{aligned}
|D| + |R| &= |D| + |R_1| + |R_2| + |R \cap O| \\
&\leq |D| + dk - \tfrac{1}{2}|S - O| + |R \cap O| \\
&= |D| + dk - \tfrac{1}{2}(|S| - |S \cap O|) + |R \cap O| \\
&\leq |D| + dk - \tfrac{1}{2}|D| + \tfrac{1}{2}|S \cap O| + |R \cap O| \quad \text{by (2)} \\
&= \tfrac{1}{2}(|D| + |S \cap O|) + |R \cap O| + dk \\
&= \tfrac{1}{2}(|D \cap O| + |D - O| + |S \cap O|) + |R \cap O| + dk \\
&\leq \tfrac{1}{2}(|D \cap O| + (|O - D| + |S \cap O|) + |S \cap O|) + |R \cap O| + dk \quad \text{by (3)} \\
&= \tfrac{1}{2}|O| + (|S \cap O| + |R \cap O|) + dk \\
&\leq \tfrac{1}{2}k + k + dk \\
&= d + \tfrac{3}{2} \quad k
\end{aligned}$$

We now give an almost-tight example for Algorithm 4 showing that its approximation ratio cannot be better than $d + 1$ (here the optimal solution deletes the two single-markers u and v instead of all $2d + 2$ single-markers):

$$\begin{array}{lccccc}
G_1 = & z_d y_d & \cdots & z_3 y_3 & z_2 y_2 & z_1 v u y_1 \\
G_2 = & z_1 y_1 & z_2 u v y_2 & z_3 y_3 & \cdots & z_d y_d \\
G_3 = & z_1 y_1 & z_2 y_2 & z_3 u v y_3 & \cdots & z_d y_d \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
G_d = & z_1 y_1 & z_2 y_2 & z_3 y_3 & \cdots & z_d u v y_d
\end{array}$$

We also have an example showing that no algorithm deleting only single-markers can achieve an approximation ratio better than d (here the optimal solution deletes one super-marker $\langle u; v \rangle$ instead of $2d$

single-markers z_i and y_i, $1 \leq i \leq d$):

$$\begin{array}{lccccc} G_1 = & z_d y_d & \cdots & z_3 y_3 & z_2 y_2 & z_1 - v - u\, y_1 \\ G_2 = & z_1 y_1 & z_2\, uv\, y_2 & z_3 y_3 & \cdots & z_d y_d \\ G_3 = & z_1 y_1 & z_2 y_2 & z_3\, uv\, y_3 & \cdots & z_d y_d \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ G_d = & z_1 y_1 & z_2 y_2 & z_3 y_3 & \cdots & z_d\, uv\, y_d \end{array}$$

Compared to the approximation upper bound of $2d$ [2, 1, 6] for the two maximization problems MSR-d and -gap-MSR-d, which almost matches (at least asymptotically) the current best lower bound of $\Omega(d/\log d)$ [6], our upper bound of $d + 1.5$ for the two minimization problems CMSR-d and -gap-CMSR-d is still far away from the constant lower bound in [6]. It is an intriguing question whether CMSR-d and -gap-CMSR-d admit approximation algorithms with constant ratios independent of d.

7 Approximation algorithm for 1-gap-CMSR-2

In this section, we give a method to transform certain approximation algorithms for 1-gap-MSR-d into approximation algorithms for 1-gap-CMSR-d, then apply this method to obtain a 2.778-approximation algorithm for 1-gap-CMSR-2.

Proposition 7.1. *If an algorithm A for 1-gap-MSR-d selects all super-markers and selects at least $1/r$ times the maximum number of single-markers selected in an optimal solution, then A finds a $(1 + (1 - 1/r)2d)$-approximation for 1-gap-CMSR-d.*

Proof. We denote by s the total number of single-markers in the input maps $G_1, \ldots, G_d$, by s^* the number of single-markers selected in an optimal solution, and by s_A the number of single-markers selected by the algorithm A. Then $s_A \geq s^*/r$. Since A does not delete any super-marker, the number k_A of markers deleted by A is equal to $s - s_A$. For 1-gap-CMSR-d, no optimal solution deletes a super-marker (otherwise the super-marker can be added back without breaking any strip since all strips have gap at most 1), so the number k^* of markers deleted by an optimal solution is equal to $s - s^*$. Since in any feasible solution, every selected single-marker must be adjacent to a deleted single-marker in some map, it follows that $s^* \leq 2dk^*$. The approximation ratio of A for 1-gap-CMSR-d is thus at most

$$\frac{k_A}{k^*} = \frac{s - s_A}{k^*} \leq \frac{(s^* + k^*) - (s^*/r)}{k^*} = 1 + \frac{(1 - 1/r)s^*}{k^*} \leq 1 + (1 - 1/r)2d. \qquad \square$$

Theorem 7.2. *There exists a 2.778-approximation for 1-gap-CMSR-2.*

The 1.8-approximation algorithm for 1-gap-MSR-2

Algorithm 5 $R(d;\ )$-approximation algorithm for -gap-MSR-d

1: $\Omega_2 \leftarrow$ the set of all candidate adjacencies of length 2 with gap $\leq$
2: $E \leftarrow$ the subset of $\Omega_2 \times \Omega_2$ of overlapping adjacencies
3: $G \leftarrow$ the $(p+1)$-claw-free graph $(\Omega_2; E)$, with $p = d(1 + \ \frac{\delta}{2}\ ) + (\ \mod 2)$ by Lemma 8.2
4: Return a $(p{=}2 + \ )$-approximation of a MAXIMUM INDEPENDENT SET of G.

Note that the values of $R(d;\ )$ for small are the following:

$$R(d;\ ) = \begin{cases} 0{:}75d + 0{:}75 + & \text{for } = 1 \\ 1{:}5d + & \text{for } = 2 \\ 1{:}5d + 0{:}75 + & \text{for } = 3 \\ 2{:}25d + & \text{for } = 4{:} \end{cases}$$

Thus Algorithm 5 improves the $2d$-approximation from [2, 6] for ≤ 3, but does not improve the $1{:}8$-approximation from [1] for $d = 2$ and $= 1$.

Algorithm 5 uses the notions of *candidate adjacencies* which are pairs of markers $\langle u; v\rangle$ such that v is a candidate successor of u, and *overlapping adjacencies* which are pairs of candidate adjacencies $\langle u; v\rangle$ and $\langle u'; v'\rangle$ such that the sets $\{u; v\} \cup S^i(u; v)$ and $\{u'; v'\} \cup S^i(u'; v')$ intersect for some i. Finally, the algorithm relies on the study of independent sets on graphs with bounded claw-size [3]; a graph has a p-claw if a vertex (the *center* of the claw) has p independent neighbors.

Lemma 8.2. *The size of any claw in the graph G created by the algorithm is upper-bounded by $p = d(1 + \ \frac{\delta}{2}\ ) + (\ \mod 2)$.*

Proof. Suppose there is a q-claw in $G = (\Omega_2; E)$. We denote by $c = \langle u; v\rangle$ its center, and by N the set of q neighbors of c in this claw: the candidate adjacency c overlaps with each $n \in N$, but adjacencies in N are pairwise non-overlapping. We partition N into $N = N_\cap \cup N_1 \cup \cdots \cup N_d$, as follows: if $n \in N$ shares a marker with c, then $n \in N_\cap$; otherwise, choose a map G_i for which $\{u; v\} \cup S^i(u; v)$ and $\{u'; v'\} \cup S^i(u'; v')$ intersect, and add n to N_i.

First note that $0 \leq |N_\cap| \leq 2$: a candidate adjacency in $N_\cap$ either contains u or v, and non-overlapping adjacencies cannot both contain u or v. Now for $1 \leq i \leq d$, we give an upper bound on $|N_i|$, depending on $|N_\cap|$. We can assume without loss of generality that u and v appear in positive form in G_i, and that N_i consists of h candidate adjacencies $\langle x_1; y_1\rangle; \cdots; \langle x_h; y_h\rangle$ appearing in this order in G_i (i.e., $\langle x_1; y_1; x_2; y_2; \cdots; x_h; y_h\rangle$ is a subsequence of G_i).

If $|N_\cap| = 2$, then x_1 must appear after u and y_h must appear before v in G_1: otherwise $\langle x_1; y_1\rangle$ (respectively $\langle x_h; y_h\rangle$) would overlap with some candidate adjacency of $N_\cap$. Since there can be at most markers between u and v, we have $2h \leq$.

If $|N_\cap| = 1$, suppose the candidate adjacency in $N_\cap$ contains u: then x_1 must appear after u in G_i. Also, x_h must appear before v, otherwise $\langle x_h; y_h\rangle$ would not overlap with $\langle u; v\rangle$. Hence we have $2h - 1 \leq$.

Finally, if $|N_\cap| = 0$, then y_1 must appear after u and x_h before v, hence $2h - 2 \leq$.

To summarize, we have the following bounds on $h = |N_i|$:

- If $|N_\cap| = 0$, then $|N_i| \leq 1 + \ \frac{\delta}{2}$
- If $|N_\cap| = 1$, and is odd, then $|N_i| \leq 1 + \ \frac{\delta}{2}$
- If $|N_\cap| = 1$, and is even, then $|N_i| \leq \ \frac{\delta}{2}$
- If $|N_\cap| = 2$, then $|N_i| \leq \ \frac{\delta}{2}$

Summing over all N_i and $N_\cap$, we have

$$\begin{aligned} q = |N| &\leq \max \begin{cases} 0 + d(1 + \lfloor \frac{\delta}{2} \rfloor) \\ 1 + d((\mod 2) + \lfloor \frac{\delta}{2} \rfloor) \\ 2 + d(\lfloor \frac{\delta}{2} \rfloor) \end{cases} \\ &= \max \{ d; 1 + d(\mod 2); 2 \} + d\lfloor \frac{\delta}{2} \rfloor \\ &= d + (\mod 2) + d\lfloor \frac{\delta}{2} \rfloor \\ &= d(1 + \lfloor \frac{\delta}{2} \rfloor) + (\mod 2) = p; \end{aligned}$$

as desired. □

We now bound the approximation ratio of Algorithm 5. If O is an optimal solution of size ℓ, there is a solution of size $\frac{2}{3}\ell$ in Ω_2: all strips in O can be decomposed into strips of length 2 or 3 with gap at most . If we remove the last element of each length-3 strip of O, we obtain a feasible solution O' of size at least $\frac{2}{3}\ell$, and whose strips correspond to candidate adjacencies appearing in Ω_2. Moreover, these strips form an independent set of $G = (\Omega_2; E)$. Using the $(p{=}2 +)$-approximation of MAXIMUM INDEPENDENT SET on $(p+1)$-claw-free-graphs given in [3], we obtain an independent set of G corresponding to a set of strips of total length $\frac{1}{p/2+\epsilon}\frac{2}{3}\ell$. This leads to an approximation ratio of $\frac{3}{4}p + $, where $p = d(1 + \frac{\delta}{2}) + (\mod 2)$.

References

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

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

[3] M. M. Halldórsson. Approximating discrete collections via local improvements. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'95)*, pages 160–169, 1995.

[4] 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*, doi:10.1007/s10878-010-9366-y.

[5] M. Jiang. On the parameterized complexity of some optimization problems related to multiple-interval graphs. *Theoretical Computer Science*, 411:4253–4262, 2010.

[6] M. Jiang. Inapproximability of maximal strip recovery. *Theoretical Computer Science*, 412:3759–3774, 2011.

[7] G. Lin, R. Goebel, Z. Li, and L. Wang. An improved approximation algorithm for the complementary maximal strip recovery problem. *Journal of Computer and System Sciences*, 78:720–730, 2012.

[8] L. Wang and B. Zhu. On the tractability of maximal strip recovery. *Journal of Computational Biology*, 17:907–914, 2010. Erratum in *Journal of Computational Biology*, 18:129, 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.

[10] B. Zhu. Efficient exact and approximate algorithms for the complement of maximal strip recovery. In *Proceedings of the 6th International Conference on Algorithmic Aspects in Information and Management (AAIM'10)*, LNCS 6124, pages 325-333, 2010.