Minimum Common String Partition Parameterized by Partition Size is Fixed-Parameter Tractable

Laurent Bulteau[1⋆] and Christian Komusiewicz[2⋆⋆]

[1] Université de Nantes, LINA - UMR CNRS 6241, Nantes, France.
[2] Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany

Abstract. The NP-hard Minimum Common String Partition problem asks whether two strings x and y can each be partitioned into at most k substrings, called blocks, such that both partitions use exactly the same blocks in a different order. We present the first fixed-parameter algorithm for Minimum Common String Partition using only parameter k.

1 Introduction

Computing the evolutionary distance between two genomes is a fundamental problem in comparative genomics [5]. Herein, the genomes are usually represented as either strings or permutations and the task is to determine how many operations of a certain kind are needed to transform one genome into the other. If the input is a pair of permutations, these problems can be formulated as sorting problems, such as Sorting by Transpositions [2] and Sorting by Reversals [1]. In this work, we study a problem in this context whose input is a pair of strings x and y. Informally, the operation to transfer x into y is to cut x into nonoverlapping substrings and to reorder these substrings such that the concatenation of the reordered substrings is exactly y. This transformation is formalized by the notion of *common string partition* (CSP): a partition $\mathcal{P}$ of two strings x and y into *blocks* $x_1x_2\cdots x_k$ and $y_1y_2\cdots y_k$ is a common string partition if there is a bijection M between $\{x_i \mid 1 \leq i \leq k\}$ and $\{y_i \mid 1 \leq i \leq k\}$ such that x_i is the same string as $M(x_i)$ for all $1 \leq i \leq k$ (see Figure 1 for an example). Herein, k is called the *size* of the common string partition $\mathcal{P}$. We study the problem of finding a minimum-size CSP:

Minimum Common String Partition (MCSP)
Input: Two strings x and y of length n, and an integer k.
Question: Is there a common string partition (CSP) $\mathcal{P}$ of size at most k of x and y?

MCSP was introduced independently by Chen et al. [3] and Swenson et al. [10] (who call the problem Sequence Cover). MCSP is NP-hard and APX-hard even when each letter occurs at most twice [7]. Damaschke [4] initiated the study of MCSP in the context of parameterized algorithmics by showing that MCSP is fixed-parameter tractable with respect to the combined parameter "partition size k and repetition number r of the input strings". Subsequently, Jiang et al. [8] showed that MCSP can be solved in $(d!)^k \cdot \text{poly}(n)$ time, where d is the maximum number of occurrences of any letter in either input string. MCSP can be solved in $2^n \cdot \text{poly}(n)$ time [6]. A greedy heuristic for MCSP was presented by Shapira and Storer [9]. In this work, we answer an open question [4, 6, 8] by showing that MCSP is fixed-parameter tractable when parameterized only by k, that is, we present an algorithm with running time $f(k) \cdot \text{poly}(n)$.

Basic Notation. A *marker* is an occurrence of a letter at a specific position in a string; we denote the marker at position i in a string x by $x[i]$. For all i, $1 \leq i < n$, the markers $x[i]$ and $x[i+1]$ are called *consecutive*. An *adjacency* is a pair of consecutive markers. An *interval* is a set of consecutive markers, that is, an interval is a set $\{x[i], x[i+1], \ldots, x[j]\}$ for some $i \leq j$. We write $[a, b]$ to denote the interval whose first marker is a and whose last marker is b. The *length* $\|I\|$

⋆ Partially supported by a DAAD scholarship.
⋆⋆ Partially supported by a post-doc scholarship of the region "Pays de la Loire".

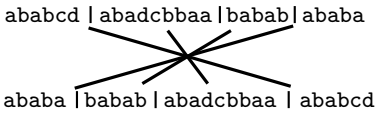

Fig. 1. An instance of MCSP with a common string partition of size four.

of an interval I is the number of markers it contains. Given two markers a and b in the same string x, we write $\overline{ab}$ to denote the signed distance between a and b, that is, $\overline{ab} = \|[a, b]\| - 1$ if a appears before b in x, and $\overline{ab} = -\|[b, a]\| + 1$, otherwise. Given two intervals s and t, we write $s \equiv t$ if they represent the same string of letters (if they have the same contents) and $s = t$ if they are the same interval, that is, they start and end at the same position in the same string. Similarly, for two markers a and b we write $a \equiv b$ if their letters are the same, and $a = b$ if the markers are identical. We say that a string s has *period* π if $s = \rho\pi^i\tau$, where $i \geq 1$, ρ is a (possibly empty) suffix of π, and τ is a (possibly empty) prefix of π. We define *offset* operators $\triangleright$ and $\triangleleft$: For each marker e and integer d, $e' = e \triangleright d$ is the marker such that $\overline{ee'} = d$, and $e \triangleleft d := e \triangleright (-d)$.

2 Fundamental Definitions and an Outline of the Algorithm.

In this section, we first present the most fundamental definitions used by our algorithm and then give a brief outline of the main algorithmic strategy followed by the algorithm.

Some Fundamental Definitions. Let $\mathcal{P} = \{x_1x_2 \dots x_\ell; y_1y_2 \dots y_\ell; M\}$ be a CSP of strings x and y. A *breakpoint* of $\mathcal{P}$ is an adjacency in x (or y) that contains the last marker of some block x_i (y_i) and the first marker of the next block x_{i+1} (y_{i+1}). We say that $\mathcal{P}$ *matches two blocks* x_i and y_j if $M(x_i) = y_j$. Furthermore, we say that $\mathcal{P}$ *matches two markers* a and b if a and b are at the same position in matched blocks. By the definition of a CSP, this implies $a \equiv b$.

The algorithm works on subdivisions of both strings into shorter parts. These subdivisions are formalized as follows.

Definition 1. *A* splitting *of a string (or an interval) z is a list of intervals $[a_1, b_1]$, $[a_2, b_2]$, …, $[a_m, b_m]$, each of length at least two, called* pieces *such that $a_1 = z[1]$, $a_{j+1} = b_j$ for all $j < m$, and $b_m = z[\|z\|]$.*

Informally, a splitting is a partition of the adjacencies of a string (or an interval) such that each part contains only consecutive adjacencies.

The strategy of the algorithm is to infer more and more information about a small CSP. To put it another way, it makes more and more restrictions on the CSP that it tries to construct. To this end, the algorithm will annotate splittings as follows: a piece is called *fragile* if it contains at least one breakpoint, and *solid* if it contains no breakpoint. To simplify the representation, the algorithm sometimes *merges* consecutive pieces $[a_i, b_i]$ and $[a_{i+1}, b_{i+1}]$ (where $b_i = a_{i+1}$) into one, that is, it removes $[a_i, b_i]$ and $[a_{i+1}, b_{i+1}]$ from some splitting and adds the interval $[a_i, b_{i+1}]$ to this splitting.

To further restrict the CSP, the algorithm finds pairs of solid pieces in x and y that are contained in blocks that are matched by the CSP. Accordingly, a pair of solid pieces s in x and t in y is called *matched* in a CSP $\mathcal{P}$ if s is contained in a block of $\mathcal{P}$ that is matched to a block that contains t. Note that matched solid pieces may correspond to different parts of their blocks. For example, one piece may contain the first marker but not the last marker of its block in x and it can be matched to a solid piece that contains the last but not the first marker of its block in y. Hence, when looking at the two blocks containing the pieces, there can be a "shift" between the matched pieces. We formalize this as follows, see Fig. 2 (left) for an example.

Definition 2. *Let $[a, b]$ be a piece of a splitting of x and $[c, d]$ be a piece of a splitting of y. The* alignment *of $[a, b]$ and $[c, d]$ of* shift δ *is the pair of* reference markers *a and $c \triangleright \delta$, where*

- $(-\overline{ab}) \leq \delta \leq \overline{cd}$,
- $[a, b] \equiv [c \triangleright \delta, c \triangleright (\overline{ab} + \delta)]$ *and* $[c, d] \equiv [a \triangleright (-\delta), a \triangleright (\overline{cd} - \delta)]$.

Hence, an alignment fixes how the interval $[a, b]$ is shifted with respect to $[c, d]$ in the matched blocks that contain the intervals. That is, if $[a, b]$ starts at position j in its block, then $[c, d]$ starts at position $j - \delta$. For matched solid pieces, an alignment thus fixes which markers are matched to each other by the CSP. In particular, the marker a is matched to $c \triangleright \delta$ and c is matched to $a \triangleleft \delta$. Note that the maximum and minimum values allowed for δ ensure that there is at least one marker in $[a, b]$ that is matched to a marker in $[c, d]$ by a CSP corresponding to this alignment. The algorithm will only consider such alignments between matched solid pieces. The second condition verifies that all pairs of matched markers indeed correspond to the same letter. Clearly, this restriction is fulfilled by every CSP that does not put breakpoints in the solid pieces $[a, b]$ and $[c, d]$. A pair of matched solid pieces is called *fixed* if it is associated with an alignment (equivalently, with a pair of reference markers) and *repetitive* otherwise (the reason for choosing this term will be given below). For a fixed solid piece s, we use s^* as shorthand for the uniquely determined reference marker of the alignment of s which is in the same string as s.

These restrictions on a possible CSP are summarized in the notion of constraints, defined as follows, see Fig. 2 (right) for an example.

Definition 3. *A* constraint $\mathcal{C}$ *is a tuple (S, F, M, R_S) such that:*

- *S is a set of solid pieces. Let S_x (S_y) denote the pieces of S from x (y).*
- *F is a set of fragile pieces. Let F_x (F_y) denote the pieces of F from x (y).*
- *The pieces of $S_x \cup F_x$ ($S_y \cup F_y$) form a splitting of x (y) in which solid and fragile pieces alternate.*
- *$M : S_x \to S_y$ is a* matching*, that is, a bijection between S_x and S_y. As shorthand, we write $s' = M(s)$ if $s \in S_x$ and $s' = M^{-1}(s)$ if $s \in S_y$.*
- *R_S is a set of alignments that contains for each matched pair of solid pieces at most one alignment.*

Our algorithm will search for CSPs that satisfy such constraints.

Definition 4. *A CSP $\mathcal{P}$* satisfies *the constraint $\mathcal{C} = (S, F, M, R_S)$ if:*

1. *All breakpoints of $\mathcal{P}$ are contained in fragile pieces.*
2. *Each fragile piece contains at least one breakpoint from $\mathcal{P}$.*
3. *Matched solid pieces are contained in matched blocks in $\mathcal{P}$.*
4. *If s is a fixed solid piece, then markers s^* and s'^* are matched in $\mathcal{P}$.*
5. *If s is a repetitive solid piece, then s, s' and the blocks containing them in $\mathcal{P}$ all have the same shortest period.*

Equivalent formulations of Conditions 1 and 2 are that (1') all solid pieces are contained in blocks of $\mathcal{P}$, and (2') different solid pieces in the same string are in different blocks. Given a CSP $\mathcal{P}$ that satisfies a constraint $\mathcal{C}$, we call a block *short*, or *undiscovered by $\mathcal{C}$*, if it does not contain a solid piece (equivalently, if it is contained in a fragile piece). The other blocks are called *long* or *discovered by $\mathcal{C}$*.

Finally, we introduce the following notion that concerns reference markers and fixed solid pieces.

Definition 5. *Let s and s' be fixed matched solid pieces in x and y. Two markers a in x and b in y are* equidistant from *s if $\overline{s^*a} = \overline{s'^*b}$. Similarly, two intervals $[a, b]$ in x and $[c, d]$ in y are* equidistant from *s if a and c are equidistant from s and b and d are equidistant from s.*

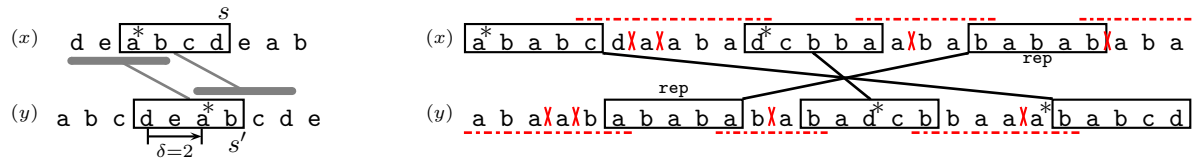

Fig. 2. Left: Example of alignment between two pieces s and s'. Reference markers are marked with a star, the shift is 2. Intervals having the same content as the pieces according to this alignment are marked in gray. Note that there also exists an alignment of shift -3, where the reference marker in y is the first occurrence of a. Right: A constraint with three pairs of solid pieces illustrated by boxes. Two of these pairs are fixed and one is repetitive (`rep`). Matched solid pieces are linked with edges. The fragile pieces (red and dashed lines) contain the breakpoints (red crosses) of a size-5 CSP satisfying the constraint.

We will use it to talk about the "local environment" of the reference markers in both strings. In particular, with this notation we can identify (sets of) markers that are matched to each other if they are both in the same block as the reference markers.

An Outline of the Algorithm and its Main Method. We now give a high-level description of the main idea of the algorithm; the pseudo-code of the main algorithm loop is shown in Algorithm 1.[3] For the discussion, assume that the instance is a yes-instance, that is, there exists a CSP $\mathcal{P}$ of size k. Since we can check in polynomial time the size and correctness of any CSP before outputting it, we can safely assume that the algorithm gives no output for no-instances; hence the focus on yes-instances. The algorithm gradually extends a constraint that is satisfied by a solution $\mathcal{P}$ and outputs $\mathcal{P}$ eventually. Initially, the constraint consists solely of two fragile pieces, one containing all of x and one all of y. We assume that the input strings are not identical. Hence, every CSP has at least one breakpoint and the initial constraint is thus satisfied by every size-k CSP.

The algorithm now aims at discovering the blocks of $\mathcal{P}$ successively, from the longest to the shortest. Recall that a block is called discovered by a constraint $\mathcal{C}$ if there is a solid piece in $\mathcal{C}$ that is contained in this block. To execute the strategy of finding shorter and shorter blocks, the algorithm needs some knowledge about the approximate (by a factor of 2) length of the longest undiscovered block in $\mathcal{P}$. To this end, the algorithm keeps and updates an integer variable β which has the following central property: Whenever there is a size-k CSP satisfying the current constraint, then there is in particular one size-k CSP $\mathcal{P}$ such that

1. the longest short block of $\mathcal{P}$ has length ℓ with $\beta \leq \ell < 2\beta$, and
2. β is minimum among all integers satisfying Property 1.

Accordingly, we call a block *β-critical* if it has length ℓ with $\beta \leq \ell < 2\beta$. To obtain β, we consider all subsets Π' of the set Π containing all powers of 2 that are smaller than n. One of these sets will contain the "correct" approximate block lengths. The central strategy is: Set β to be the largest value in Π'. Discover all β-critical blocks. Then, there is a satisfying CSP such that all undiscovered blocks are shorter than the current β. Thus update β by taking the next largest value from Π'. Then, again discover all β-critical blocks, update β again and so on.

First, note that there is at least one block of length at least $\lceil n/k \rceil$ since $\mathcal{P}$ has size k, so $\max \Pi' \geq \lceil n/2k \rceil$. Furthermore, for any CSP of size k, $|\Pi'| \leq k$. Hence, the outer algorithm loop of Algorithm 1 is traversed once for the correct Π'. Note furthermore, that the number of subsets of Π is $O(2^{\log n}) = O(n)$. Hence, there are $O(n)$ traversals of the outer loop of the main method.

Consider now the traversal for the correct set Π'. The inner loop of the algorithm consists of two main steps. In the first step, called `split`, the algorithm discovers the β-critical blocks. More precisely, it refines $\mathcal{C}$ by breaking fragile pieces into shorter pieces (of length $\lceil \beta/3 \rceil$) and

[3] Parts of this algorithm, in particular the `split` procedure follow somewhat the approach of Damaschke [4].

Algorithm 1 The main algorithm loop `MCSP`(x, y, k).

```
1   Π := {i ∈ ℕ | i < n ∧ ∃j ∈ ℕ : 2^j = i}
2   C := {S := ∅, F := {[x[1], x[n]], [y[1], y[n]]}, M := ∅, R_S := ∅}    // initially only two fragile pieces
3   for each Π' ⊆ Π with max Π' ≥ ⌈n/2k⌉ ∧ |Π'| ≤ k :
4       β ← max Π'; Π' ← Π' ∪ {0} \ {β}    // 2-approx. length of longest block
5       repeat until β < 4 :
6           split                          // discover blocks of length at least β
7           β ← max Π'; Π' ← Π' \ {β}       // update 2-approx. length of longest undiscovered blocks
8           frames                         // reduce length of fragile pieces
9       branch into all cases to set breakpoints within fragile pieces
10      if the resulting string partition P is a size-k CSP : output P
```

identifying those that are contained in β-critical blocks. It then produces a matching and, if this is possible without considering too many options, aligns these blocks.

To be efficient `split` requires that the input fragile pieces are short enough compared to β and k. Initially, this is not a problem, since the fragile pieces have length n, and $\beta \geq n/2k$. After `split`, however, we update β. Hence, between two calls to `split` the fragile pieces have to be reduced in order to fit the undiscovered blocks more "tightly". This is the objective of `frames`, which uses a set of rules to identify smaller intervals containing all breakpoints of $\mathcal{P}$. It thus shrinks the fragile pieces of $\mathcal{C}$ so that they are sufficiently small for the next call to `split`.

The algorithm now continues with this process for smaller and smaller values of β. It stops in case $\beta < 4$, since it can then locate all breakpoints by applying a brute-force branching. Note that in order to ensure that there is always a $\beta < 4$, we add the value 0 to set Π' in Line 4 of the main method.

In the remainder of this work, we give the details for the procedures `split` and `frames`. In Section 3, we describe the `split` procedure, and show its correctness. We also show, using several properties of `frames` as a black box, our main result. Then, in Sections 4 and 5, we fill in the blanks by proving the properties of `frames`.

The algorithm is a branching algorithm that extends the constraint $\mathcal{C}$ in each branch. In order to simplify the pseudo-code somewhat, we describe the algorithm in such a way that the variables $\mathcal{C}$ and β are global variables. After a branching statement in the pseudo-code, the algorithm continues in each branch with the following line of the pseudo-code. If a branch is known to be unsuccessful, then the algorithm returns immediately to the branching statement that created this branch (or to the branching statement above, if the current branch is the last branch of that statement). We denote this by the "abort branch" command; all modifications within this branch are undone.

3 Splitting of Fragile Pieces

In this section, we describe the procedure `split` and show its correctness. The pseudo-code of `split` is shown in Algorithm 2. At the beginning of `split` the constraint contains a set of discovered blocks. Assume that all blocks of length at least 2β are discovered by this constraint. The aim of `split` now is to perform a branching into several cases such that in at least one of the created branches the constraint $\mathcal{C}$ now additionally contains all β-critical blocks. Hence, in this branch all blocks of length at least β are discovered. Procedure `split` starts by replacing each former fragile piece by a splitting where all new pieces have length $\lceil \beta/3 \rceil$ except for the rightmost new piece of each such splitting which can be shorter. We call such a splitting a $\lceil \beta/3 \rceil$-*splitting*. It then considers all branches where each piece is either fragile or solid. In order to maintain the alternating condition, consecutive solid (resp. fragile) pieces are merged into one solid (fragile) piece, Lines 7–9.

Next, `split` extends the matching and the set of alignments of the constraint. All possible matchings are considered in separate branches (Lines 12–14). Then, `split` performs an exhaus-

Algorithm 2 Procedure `split`. Global variables: $\mathcal{C} = (S, F, M, R_S)$ and β.

```
1  N := ∅                                          // the set of new pieces
2  for each fragile piece f ∈ F :
3      F ← F \ {f}                                 // old fragile pieces are removed
4      N ← N ∪ "⌈β/3⌉-splitting of f"               // update set of new pieces
5  for each p ∈ N :                                // make p either fragile or solid
6          branch into the case that either S ← S ∪ {p} or F ← F ∪ {p}
7  while ∃ consecutive pieces s1, s2 s.t. {s1, s2} ⊆ S (or {s1, s2} ⊆ F) :
8      p := "merged interval of s1 and s2"
9      S ← (S ∪ p) \ {s1, s2} (or F ← (F ∪ p) \ {s1, s2})
10 if |Sx| ≠ |Sy| : abort branch                   // no bijection of solid pieces exists
11 if |Fx| ≥ k or |Fy| ≥ k : abort branch          // too many fragile pieces in x (or y)
12 while ∃ unmatched solid piece s ∈ Sx :
13     for each unmatched solid piece t in Sy :
14         branch into the case that M(s) := t
15 for each new pair (s, t) of matched solid pieces :
16     i := "number of alignments with shift δ s.t. |δ| ≤ ⌈β/3⌉"
17     if i ≤ 6 : for each alignment branch into the case to add this alignment to RS
18     else: branch into the cases to:             // s and s' are periodic
           – align s and s' such that l_break(s) and l_break(s') are equidistant from s
           – align s and s' such that r_break(s) and r_break(s') are equidistant from s
           – do not align s and s'
```

tive branching over all alignments for a given pair of solid pieces, but only if there are very few of them (Line 17). If there are too many (Line 18), then it can be seen that the pieces are periodic with a short period length. Thus, the blocks containing them might be periodic as well. If the blocks are not periodic, then there are at most two alignments that the algorithm needs to consider: informally, the period in the blocks can be "broken" either to the left or to the right of the pieces. To specify these two possibilities more clearly, we introduce the following notation. Let $s = [a, b]$ be an interval in a string x such that s has period π. Then, we denote by $\mathrm{l}_{\mathrm{break}}(s)$ the rightmost marker in x such that $[\mathrm{l}_{\mathrm{break}}(s), b]$ does not have period π. Similarly, let $\mathrm{r}_{\mathrm{break}}(s)$ be the leftmost marker in x such that $[a, \mathrm{r}_{\mathrm{break}}(s)]$ does not have period π. If the blocks are periodic, there may be too many possible alignments, and the alignment between the pieces will be fixed at a later point (when β becomes smaller than the period). However, the algorithm will use the "knowledge" that the blocks are periodic in the `frames` procedure.

We now show that `split` is correct if the input constraint can be satisfied and that it discovers all β-critical blocks.

Lemma 1. *Let $\mathcal{C}$ be the constraint at the beginning of* `split`*, and let $\mathcal{P}$ be a size-k CSP satisfying $\mathcal{C}$ such that all blocks of length at least 2β of $\mathcal{P}$ are discovered by $\mathcal{C}$. Then,* `split` *creates at least one branch whose constraint $\mathcal{C}$*

- *is satisfied by $\mathcal{P}$, and*
- *all blocks of length at least β are discovered by $\mathcal{C}$*

Proof (of Lemma 1). Let $B = \{(x^1, y^1), \ldots, (x^\ell, y^\ell)\}$ be the uniquely defined set of matched pairs of undiscovered blocks in $\mathcal{P}$ that are β-critical.

Consider the following branching for Lines 5–6 for each piece $p \in N$: If p is contained in some block x^i or y^i of B, then branch into the case that p is added to S. Otherwise, branch into the case that p is added to F (note that we may add in F some pieces that do not contain any breakpoint, but are contained in blocks not in B).

Now consider the constraint obtained for the above branching after the merging operations performed in Lines 7–9. We show that $\mathcal{P}$ satisfies Conditions 1 and 2 of this constraint. First, consider a breakpoint in $\mathcal{P}$. This breakpoint is contained in some fragile piece f of the input constraint since $\mathcal{P}$ satisfies this input constraint. Hence, it is contained in some new piece p of the splitting of this fragile piece. Clearly, the piece p is added to F in the considered branching.

Moreover, in case Lines 7–9 merge fragile pieces, the resulting piece is also fragile, hence p remains in a fragile piece. Consequently, all breakpoints of $\mathcal{P}$ are in fragile pieces of F, and thus Condition 1 is satisfied by $\mathcal{P}$.

Now consider a fragile piece $f \in F$ after Lines 7–9 of the algorithm. We show that f contains at least one breakpoint. Note that f is obtained after a (possibly empty) series of merging operations. After the merging, f is between two solid pieces. If f is also a fragile piece in the input constraint, then f contains a breakpoint since $\mathcal{P}$ satisfies the input constraint. Otherwise, f is contained in a fragile piece of the input constraint, and at least one of its neighbor pieces is a new solid piece s. Since f (or all the smaller pieces that were merged to f) are added to F by the branching, they are not contained in the block that contains s. Hence, f contains the breakpoint between the first (or last) marker of the block containing the new solid piece and its predecessor (or successor). Thus, Condition 2 is also satisfied by $\mathcal{P}$.

Note that the above also implies that, for each x^i of B, there is exactly one new solid piece that is contained in x^i. Similarly, for each y^i of B, there is exactly one new solid piece that is contained in y^i. Note that in this branching, $|S_x| = |S_y|$ and furthermore, since $\mathcal{P}$ has size k, $|F_x| < k$ and $|F_y| < k$. Hence, the algorithm does not abort in Lines 10 and 11. We now consider the branching in which for each pair (x^i, y^i), the two corresponding solid pieces are matched to each other. Clearly, this branching fulfills Condition 3: the condition holds obviously for all pieces contained in blocks of B. Furthermore, it holds for all old solid pieces since for these, the matching M has not changed. Note that the function M also remains a bijection: it is changed only for unmatched solid pieces, and the number of new solid pieces in x and y is equal.

It remains to show that there is a branching in which Conditions 4 and 5 also hold. Consider a pair of matched solid pieces s and s', and the blocks x^i, y^i containing them. We use the following technical claim in order to clarify the discussion; it will be proven afterwards.

> *Fact.* If there are more than six alignments of s and s' whose shift have an absolute value of at most $\lceil \beta/3 \rceil$, then
>
> i. s and s' are periodic with the same shortest period π (with $\|\pi\| \leq \lceil \beta/3 \rceil/2$);
> ii. if the blocks x^i and y^i do not have period π, then in $\mathcal{P}$ either $\mathrm{l}_{\mathrm{break}}(s)$ is matched to $\mathrm{l}_{\mathrm{break}}(s')$, or $\mathrm{r}_{\mathrm{break}}(s)$ is matched to $\mathrm{r}_{\mathrm{break}}(s')$ (or both).

Let a be the leftmost marker of s and $\tilde{a}$ be the marker matched to a in $\mathcal{P}$. Then $\{a, \tilde{a}\}$ is an alignment for (s, s') whose shift has an absolute value less than $\lceil \beta/3 \rceil$: there are at most $\lceil \beta/3 \rceil - 1$ markers preceding either s or s' that can belong to the same block since the pieces of the $\lceil \beta/3 \rceil$-splitting preceding s and s' are fragile and thus not contained in the same blocks. If the condition of Line 17 is satisfied, then there is one branch where alignment $\{a, a'\}$ is added to R_S. Otherwise, by the fact above, the following cases are possible. Either $\{a, a'\}$ is one of the alignments where $\mathrm{l}_{\mathrm{break}}(s)$ is matched to $\mathrm{l}_{\mathrm{break}}(s')$ or $\mathrm{r}_{\mathrm{break}}(s)$ is matched to $\mathrm{r}_{\mathrm{break}}(s')$, in which cases $\{a, a'\}$ is added to R_S in one of the branches. Otherwise, (s, s') is not fixed, and s and s' are contained in blocks having the same shortest periods.

Altogether this shows the first claim of the lemma. The second claim can be seen as follows. The blocks of length $\ell \geq 2\beta$ are already discovered, and the corresponding solid pieces remain in the constraint. It thus remains to consider the β-critical blocks. We show that for each x^i there is at least one piece that is contained in x^i. Consider the marker a at position $\lceil \beta/3 \rceil$ in x^i and a piece s of the $\lceil \beta/3 \rceil$-splitting that contains this marker. Then s contains only markers from x^i since s has length at most $\lceil \beta/3 \rceil$ and x^i has length at least $\beta \geq 2\lceil \beta/3 \rceil$ (for $\beta \geq 4$). Afterwards, s is only merged with other pieces that are contained in x^i (recall that in the considered branching there is a fragile piece between all solid pieces from different blocks). Hence, the second claim of the lemma also holds.

It remains to show the correctness of the claimed fact. We first need to prove the following claim. Define the $\lceil \beta/3 \rceil$-*middle* of an interval $[u, v]$ as the length-$\lceil \beta/3 \rceil$ interval centered in $[u, v]$

(formally, the interval $[\hat{u}, \hat{v}]$ with $\hat{u} = u \triangleright \lfloor(\overline{uv} - \lceil\beta/3\rceil)/2\rfloor$ and $\hat{v} = v \triangleleft \lceil(\overline{uv} - \lceil\beta/3\rceil)/2\rceil$). Then s contains the $\lceil\beta/3\rceil$-middle of x^i and s' contains the $\lceil\beta/3\rceil$-middle of y^i.

The claim is shown for $s = [a, b]$ (the proof for s' is similar). Write $x^i = [u, v]$, and $[\hat{u}, \hat{v}]$ the $\lceil\beta/3\rceil$-middle of x^i. First note that since x^i has length at least β, we have $\overline{uv} \geq \beta - 1$. We show that a is in the interval $[u, \hat{u}]$:

$$\begin{aligned}
\overline{u\hat{u}} &= \lfloor(\overline{uv} - \lceil\beta/3\rceil)/2\rfloor \\
&\geq \lfloor(\beta - 1 - \lceil\beta/3\rceil)/2\rfloor \\
&\geq \lfloor(\lfloor 2\beta/3\rfloor - 1)/2\rfloor \\
&\geq \lfloor(2\beta/3 - 1.7)/2\rfloor \\
&\geq \lfloor\beta/3 - 0.85\rfloor \\
&\geq \lceil\beta/3\rceil - 2.
\end{aligned}$$

Since the piece with right endpoint a in the $\lceil\beta/3\rceil$-splitting is fragile (it has not been merged with s), it contains a breakpoint of $\mathcal{P}$ and hence a marker strictly to the left of u. Moreover it has length at most $\lceil\beta/3\rceil$, so $\overline{ua} \leq \lceil\beta/3\rceil - 2$, which implies that a is in the interval $[u, \hat{u}]$. Similarly, b is in the interval $[\hat{v}, v]$, and $[a, b]$ contains the $\lceil\beta/3\rceil$-middle of x^i.

We can now turn to proving the two statements of the fact.

(i) Let $s = [a, b]$, $s' = [a', b']$ and $\delta_1, \delta_2, \ldots, \delta_m$ be the shifts of the $m \geq 7$ alignments such that $-\lceil\beta/3\rceil \leq \delta_1 \leq \delta_2 \leq \ldots \leq \delta_m \leq \lceil\beta/3\rceil$. Write i the index such that $\delta_{i+1} - \delta_i$ is minimal, and $p = \delta_{i+1} - \delta_i$. We thus have

$$\begin{aligned}
p &\leq \frac{2\lceil\beta/3\rceil}{m-1} \\
&\leq \lceil\beta/3\rceil/2
\end{aligned}$$

Recall also that both s and s' have length at least $2\lceil\beta/3\rceil - 1$. Let q be an integer with $p \leq q \leq \overline{ab}$. Using the second condition in the definition of alignment, we have

$$\begin{aligned}
a \triangleright q &\equiv a' \triangleright (\delta_i + q) \quad (\text{since } a \triangleright q \in [a, b]) \\
&= a' \triangleright (\delta_{i+1} + q - p) \\
&\equiv a \triangleright (q - p) \quad (\text{since } a \triangleright (q - p) \in [a, b])
\end{aligned}$$

Thus intervals $[a, b]$ and (symmetrically) $[a', b']$ are both periodic with period length p: the shortest periods of s and s' have length at most $\lceil\beta/3\rceil/2$.

Using the fact that s and s' both contain the $\lceil\beta/3\rceil$-middle of the matched blocks in which they are contained, they have a common substring of length greater than twice their shortest periods. They thus have the same shortest period.

(ii) Recall that x^i (y^i) is the block containing s (s') in $\mathcal{P}$. Write $[\hat{u}, \hat{v}]$ ($[\hat{u}', \hat{v}']$) the $\lceil\beta/3\rceil$-middle of x^i (y^i). Since $[\hat{u}, \hat{v}] \subset s$ and $\|[\hat{u}, \hat{v}]\| \geq \|\pi\|$, we have that $\mathrm{l}_{\mathrm{break}}(s)$ is the rightmost marker in x and $\mathrm{l}_{\mathrm{break}}(s)$ is the leftmost marker in x such that intervals $[\mathrm{l}_{\mathrm{break}}(s), \hat{v}]$ and $[\hat{u}, \mathrm{r}_{\mathrm{break}}(s)]$ do not have period π. We have the similar property for $\mathrm{l}_{\mathrm{break}}(s')$ ($\mathrm{r}_{\mathrm{break}}(s')$) and $\hat{v}'$ ($\hat{u}'$).

Since x^i contains $[\hat{u}, \hat{v}]$, then either x_i has period π, either it contains $\mathrm{l}_{\mathrm{break}}(s)$ or $\mathrm{r}_{\mathrm{break}}(s)$. Suppose that x^i contains $\mathrm{l}_{\mathrm{break}}(s)$ (the case where x^i contains $\mathrm{r}_{\mathrm{break}}(s)$ is similar). Write l' the marker in y^i matched to $\mathrm{l}_{\mathrm{break}}(s)$ by $\mathcal{P}$. Then $[l', \hat{v}'] \equiv [\mathrm{l}_{\mathrm{break}}(s), \hat{v}]$ does not have period π, and for all $m' \in [l' \triangleright 1, \hat{v}']$, $[m', \hat{v}']$ has period π. Thus, l' is the rightmost marker such that $[l', \hat{v}']$ does not have period π, and $l' = \mathrm{l}_{\mathrm{break}}(s')$: markers $\mathrm{l}_{\mathrm{break}}(s)$ and $\mathrm{l}_{\mathrm{break}}(s')$ are matched in $\mathcal{P}$. □

The following trivial observation follows from the check in Line 11 of `split`. It is useful for bounding the running time of `split` (in particular for later calls to `split`).

Observation 1 *After* `split` *has finished, the constraint contains at most* $2k-2$ *fragile pieces from each of* x *and* y*. The overall number of solid pieces is thus at most* $2k$*.*

To obtain a fixed-parameter algorithm for parameter k, we now "shrink" the fragile pieces between the solid pieces of the constraint. This will ensure that in the next call to `split`, the number of new pieces created in the splitting will be bounded by a function of k. Note that by Lemma 1, `split` has discovered *all* pieces that have length at least β. Hence, we now update the value β denoting the approximate length of the longest short blocks (by taking the largest remaining value from Π). Then, `frames` uses this updated value of β to shrink the fragile pieces. For the moment, we make some claims about `frames`; their proof is deferred to the Sections 4 and 5. First, we claim that `frames` is correct, that is, there is at least one good branching for yes-instances.

Lemma 2. *If there exists a size-*k *CSP* $\mathcal{P}$ *satisfying* $\mathcal{C}$ *at the beginning of* `frames` *such that longest undiscovered block is* β*-critical, then* `frames` *creates at least one branch such that the constraint in this branch is satisfied by a size-*k *CSP* $\mathcal{P}'$ *whose longest undiscovered block has length at most* $2\beta-1$*.*

Second, `frames` increases the exponential part of the running time by a factor that depends only on k.

Lemma 3. *Overall, the calls to* `frames` *create* $(2k)^{4k^2} \cdot k^{O(k)}$ *branches; all other parts of the algorithm can be performed in* $\mathrm{poly}(n)$ *time.*

Finally, to bound the number of branches in the subsequent call to `split`, and for the case $\beta < 4$, we use the following lemma.

Lemma 4. *When* `frames` *terminates, every fragile piece has length at most* $12(k^2+k)k\beta$*.*

Note that the above also holds before the first call of `split`. Using these lemmas, we obtain our main result.

Theorem 1. Minimum Common String Partition *can be solved in* $k^{21k^2}\,\mathrm{poly}(n)$ *time; it is thus fixed-parameter tractable with respect to the partition size* k*.*

Proof (of Theorem 1). For the correctness proof assume that the instance is a yes-instance (for a no-instance the algorithm can always check the correctness and size of a CSP before returning, thus it has empty output for no-instances). Then, assuming that the input strings are not identical, there is a CSP $\mathcal{P}$ satisfying the initial constraint $\mathcal{C}$ which demands only that there is at least one breakpoint in x and in y.

We now show that there is a set Π' of powers of 2, all of which are smaller than n such that the algorithm outputs, in at least one of its branches, a size-k CSP, in case the main algorithm loop is traversed for this set Π'.

Let β be the smallest integer such that there is a size-k CSP in which the longest block is β-critical. Then, the largest integer of Π' is β. Now, if $\beta < 4$ the algorithm directly finds all breakpoints by a brute-force branching. Otherwise, the procedure `split` is called. By Lemma 1, this procedure creates at least one branch where the constraint is satisfied by some size-k CSP $\mathcal{P}$ and all its blocks of length at least β are discovered by $\mathcal{C}$. Consider an arbitrary branch with this property. Now, let β denote the smallest power of 2 such that there is a CSP satisfying the current constraint $\mathcal{C}$ in which the longest blocks are β-critical. This integer β is the second largest integer of Π'. The algorithm now calls `frames` and by Lemma 2 obtains in at least one branch a constraint such that there is a size-k CSP that satisfies the constraint in this branch. Furthermore, also by Lemma 2 the longest undiscovered block in this CSP has length at most $2\beta-1$. By the choice of β, it follows that the longest undiscovered block of this CSP is β-critical. Now, the algorithm either finds all breakpoints by brute-force (if $\beta < 4$)

or again calls the procedure `split` to discover all β-critical blocks. This whole procedure is repeated for smaller and smaller β, each time β is defined as the smallest power of two such that there is a size-k CSP satisfying the current constraint $\mathcal{C}$ whose longest undiscovered block is β-critical. The set Π' contains exactly all integers obtained this way. Eventually, $\beta < 4$ and the algorithm branches by brute-force into all cases to set the breakpoints without violating the current constraint. Clearly, one of these cases is equivalent to a CSP satisfying this constraint. The algorithm verifies that this is indeed a CSP and that it has size at most k and correctly outputs the CSP. Hence, the algorithm is correct.

It remains to show the running time of the algorithm. First, the for-each-loop in the main method is executed $O(2^{\log n}) = O(n)$ times. Second, by the restriction on Π', the repeat-loop in the main method is executed at most k times. To obtain the claimed running time, we bound the number of branches created in each call to `split`.

In each call to `split` the total length of the fragile pieces is less than $(2k)12(k^2+k)k\beta = 24(k^4+k^3)\beta$: In the first call, $\beta > n/2k$, so the bound holds. In the other cases, there are, by Observation 1 at most $2k-2$ fragile pieces in x and y. Furthermore, in this case `split` is called after `frames`. Thus, by Lemma 4, each fragile piece has length at most $12(k^2+k)k\beta$, and the overall bound follows.

The procedure splits the fragile pieces into new pieces of length at most $\lceil \beta/3 \rceil$ (i.e. there is a distance $\lceil \beta/3 \rceil - 1$ between the left endpoints of two consecutive pieces of the same splitting). Since $\beta \geq 4$, we have $\lceil \beta/3 \rceil - 1 \geq \beta/6$. Hence, this creates less than $144(k^4+k^3)$ new pieces of length $\lceil \beta/3 \rceil$ plus at most one additional shorter piece at the end of each fragile piece. Hence, $145k^4$ is an upper bound on the number of new pieces. Branching for each piece into the case that it is solid or fragile can be done in 2^{145k^4} branches. The number of necessary branches for this part of `split` can be reduced as follows: Since we merge series of consecutive pieces in F or S, and since we do not need to consider branches with more than k solid pieces, we can directly look for the first and last piece of each β-critical block. This creates $O(\binom{145k^4}{4k}) = O(\frac{(145k^4)^{4k}}{(4k)!})$ branches in each call of `split`.

The matching requires up to $k!$ branches, and the alignment at most 6^k. Since $145^{4k}k!6^k = o((4k)!)$, we can bound the number of branches in each call of `split` by k^{16k}. The `split` procedure is called at most k times (by the restriction on Π), thus creating $O(\binom{k^{16k}}{k}) = O(k^{16k^2})$ branches throughout the algorithm. Finally, the number of branches created in `frames` is $(2k)^{4k^2} \cdot k^{O(k)}$ by Lemma 3, and the number of branches created in the final brute-force can be bounded as follows. The length of the fragile pieces is $O(k^4+k^3)$ and we need to guess at most $2k-2$ precise breakpoint positions from this number. This can be done with $k^{O(k)}$ branches.

Finally, note that all other steps of the algorithm can be clearly performed in polynomial time. Altogether, the total running time of the algorithm thus is

$$O(k^{2k}n) \cdot k^{16k^2} \cdot k^{O(k)} \cdot (2k)^{4k^2} \cdot k^{O(k)} \cdot \text{poly}(n) = k^{21k^2}\,\text{poly}(n).$$

□

4 Putting Frames Next to Fixed Pieces

In this and the next section, we prove the two claimed lemmas concerning `frames`. Informally, we show that, with the right constraint in the beginning, `frames` finds a constraint $\mathcal{C}$ that is satisfied by a size-k CSP $\mathcal{P}$ whose longest undiscovered block has length at most $2\beta - 1$. Moreover, the length of each fragile piece is $O(k^3\beta)$ in every constraint produced by `frames`. The pseudo code of `frames` is shown in Algorithm 3.

The approach of `frames` is to use a set of reduction rules to put "frames" into the fragile pieces, where a frame is an interval within the fragile piece that contains *all* breakpoints that are contained in this piece. We call the actual shortest interval containing all breakpoints of a

Algorithm 3 Procedure `frames`. Global variables: $\mathcal{C}$, β.

```
1  w := 2βk + 1                                   // upper bound on window length
2  repeat :
3      Compute the maximum extension of each solid piece,
       the piece graph G[C, Φ := ∅], and the strips of each rep−rep path
4      while there is a frameless fragile piece :
5          place frames in fragile pieces by applying Frame Rules 1–6
6      for each fragile piece : apply Fitting Rule 1
7      new-align := False                         // Fix pieces with long periods:
8      for each repetitive solid piece s (with period π_s) :
9          if all fragile pieces adjacent to s or s' have length at most 6(k^2 + k) ||π_s|| :
10             for each feasible alignment branch into the case to add this alignment to R_S
11             new-align ← True
12 until new-align = False
13 return the modified constraint C
```

fragile piece a "window", defined as follows. Let $\mathcal{P}$ be a size-k CSP satisfying $\mathcal{C}$, and let f be a fragile piece in $\mathcal{C}$. The *window* of f is the interval $[a, b]$ such that $\{a, a \triangleright 1\}$ is the leftmost breakpoint of $\mathcal{P}$ in f and $\{b \triangleleft 1, b\}$ is the rightmost breakpoint of f. Since a frame is required to contain all breakpoints of a fragile piece it can be seen as a "super"-approximation of the actual window. A formal definition of frames is as follows.

Definition 6. *Let $\mathcal{C}$ be a constraint. A* frame $[a, b]$ *for a fragile piece f of $\mathcal{C}$ is an interval that is contained in f. A* frame set *for $\mathcal{C}$ is a set Φ of frames such that each fragile piece f contains at most one frame. A CSP $\mathcal{P}$ that satisfies $\mathcal{C}$* satisfies *a frame set Φ for $\mathcal{C}$ if each breakpoint of $\mathcal{P}$ is contained in some frame of Φ.*

The approach to add the frames to the constraint can be summarized as follows: first, we compute an upper bound w on the length of the windows that only depends on β and k. Then, we apply a series of *frame rules* that eventually place a frame in all fragile pieces (Lines 4–5). As we will show, the frame length then depends on w (and thus on k and β) and on the maximum period length of the unfixed (repetitive) solid pieces. Since the frames contain all breakpoints of $\mathcal{P}$, it is possible to reduce fragile pieces until they "fit" their frames (Line 6). We now check whether there are some unfixed solid pieces with a long period compared to w. If this is the case, then the number of "feasible" alignments for these pieces is small, and we can thus branch how to align these pieces (Lines 7–11). Formally, we call an alignment of $s = [a, b]$ and $s' = [a', b']$ *feasible* for $\mathcal{C}$ if the interval equidistant to $[a, b]$ ($[a', b']$) from s does not intersect any other solid piece than s' (s) in $\mathcal{C}$. Note that each satisfying CSP can only have feasible alignments, otherwise there is at least one fragile piece without breakpoints.

Afterwards, we go back to applying the frame rules (we will obtain shorter frames since the number of fixed pieces has increased). If this is not the case, that is, all periods are short compared to w, then we show that the maximum frame length depends only on w. Hence, in this case they are sufficiently short, and the `frames` procedure has achieved its goal. The algorithm thus returns to the main method where it calls `split` to find new solid pieces.

In this section, we describe the frame rules that place frames in fragile pieces which are next to fixed pieces and some further simple frame rules. Before doing so, we define two concepts that will be used by the frame rules: *maximum extensions* and the *piece graph*. Roughly speaking, maximum extensions are used locally to bound the position of some breakpoints in the fragile pieces. The piece graph provides a structural view of the relationship between pieces and is used to show that one of the frame rules can always be applied in case there is a frameless fragile piece.

Maximum extension of solid pieces. Let s be a solid piece in a constraint $\mathcal{C}$. The *maximum extension* of s is the interval $[\mathrm{l}_{\mathrm{ext}}(s), \mathrm{r}_{\mathrm{ext}}(s)]$ containing s where $\mathrm{r}_{\mathrm{ext}}(s)$ and $\mathrm{l}_{\mathrm{ext}}(s)$ are defined as

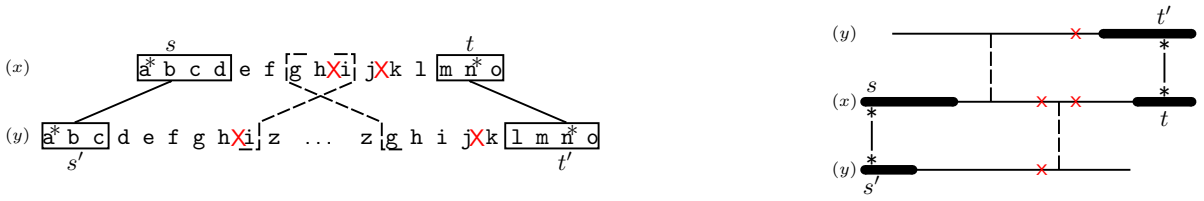

Fig. 3. Left: two pairs of fixed solid pieces (s, s') and (t, t'). Reference markers are shown with an asterisk, maximum extensions are delimited with dashed lines, and the breakpoints of some possible CSP are marked with red crosses. Right: a simplified representation of the same pieces, where thick (resp. thin) lines are used for solid (resp. fragile) pieces.

follows. If s is fixed, then let ℓ be the largest integer such that $[s^*, s^* \triangleright \ell] \equiv [s'^*, s'^* \triangleright \ell]$, and that no marker of $[s^*, s^* \triangleright \ell]$ or $[s'^*, s'^* \triangleright \ell]$ is in a solid piece other than s or s'. Then $\mathrm{r}_{\mathrm{ext}}(s) := s^* \triangleright \ell$ and $\mathrm{r}_{\mathrm{ext}}(s') := s'^* \triangleright \ell$. If s is repetitive with shortest period π_s, then let a be the leftmost marker in s and define $\mathrm{r}_{\mathrm{ext}}(s)$ as the rightmost marker such that the interval $[a, \mathrm{r}_{\mathrm{ext}}(s)]$ has period π_s, and that no marker in $[a, \mathrm{r}_{\mathrm{ext}}(s)]$ is in a solid piece other than s. Marker $\mathrm{l}_{\mathrm{ext}}(s)$ is obtained symmetrically.

The following proposition is a straightforward consequence of the definition of maximum extension.

Proposition 1. *Let s be a fixed solid piece, and let $[a, b]$ and $[c, d]$ be two intervals that are equidistant from s and such that $[a, b]$ is contained in $[\mathrm{l}_{\mathrm{ext}}(s), \mathrm{r}_{\mathrm{ext}}(s)]$ and $[c, d]$ is contained in $[\mathrm{l}_{\mathrm{ext}}(s'), \mathrm{r}_{\mathrm{ext}}(s')]$. Then, $[a, b] \equiv [c, d]$.*

Note that, as a special case, the above proposition includes single markers (that is, length-one intervals). The next proposition simply states formally that the maximum extensions of a solid piece contain the block which contains the solid piece.

Proposition 2. *Let $\mathcal{C}$ be a constraint and s be a solid piece of $\mathcal{C}$. Any CSP that satisfies $\mathcal{C}$ has a block which contains s and is contained in $[\mathrm{l}_{\mathrm{ext}}(s), \mathrm{r}_{\mathrm{ext}}(s)]$. Furthermore, let f be a fragile piece next to s. Then, the window in f contains at least one marker of $[\mathrm{l}_{\mathrm{ext}}(s), \mathrm{r}_{\mathrm{ext}}(s)]$.*

Proof (of Proposition 2). If s is fixed, then the block containing s cannot contain the marker $\mathrm{l}_{\mathrm{ext}}(s) \triangleleft 1$: if this marker is contained in the block, it is matched to $\mathrm{l}_{\mathrm{ext}}(s') \triangleleft 1$. By definition of $\mathrm{l}_{\mathrm{ext}}(s)$, either $\mathrm{l}_{\mathrm{ext}}(s) \triangleleft 1 \not\equiv \mathrm{l}_{\mathrm{ext}}(s') \triangleleft 1$ or one of $\mathrm{l}_{\mathrm{ext}}(s) \triangleleft 1$, $\mathrm{l}_{\mathrm{ext}}(s') \triangleleft 1$ belongs to a different solid piece t. In the first case, we do not obtain a CSP; in the second case, there is at least one fragile piece without a breakpoint. Similarly, the block containing s cannot contain $\mathrm{r}_{\mathrm{ext}}(s) \triangleright 1$.

Every repetitive solid piece s is contained in a block which is periodic with the same shortest period π as s. By definition of $\mathrm{l}_{\mathrm{ext}}()$ and $\mathrm{r}_{\mathrm{ext}}()$ for repetitive solid pieces this block must thus be contained in $[\mathrm{l}_{\mathrm{ext}}(s), \mathrm{r}_{\mathrm{ext}}(s)]$.

Finally, consider a fragile piece f to the right (to the left) of s. The window in f contains the last (first) marker of the block containing s. By the above it thus contains at least one marker of $[\mathrm{l}_{\mathrm{ext}}(s), \mathrm{r}_{\mathrm{ext}}(s)]$. □

The Piece Graph. Given a constraint $\mathcal{C}$ and a frame set Φ, the *piece graph* $G[\mathcal{C}, \Phi]$ is the bipartite graph $G := (V_S \cup V_F, E)$ constructed as follows.

- V_F contains one vertex v_f for each frameless fragile piece $f \in F$,
- V_S contains, for each repetitive solid piece $s \in S_x$ a vertex v_s, and for each fixed piece $s \in S_x$, two vertices l_s and r_s (for *left* and *right*).
- For a fixed solid piece s and a fragile piece $f \in F$, G contains the edge $\{v_f, l_s\}$ if the last marker of f is the first marker of s or of s', and the edge $\{v_f, r_s\}$ if the first marker of f is the last marker of s or of s'.

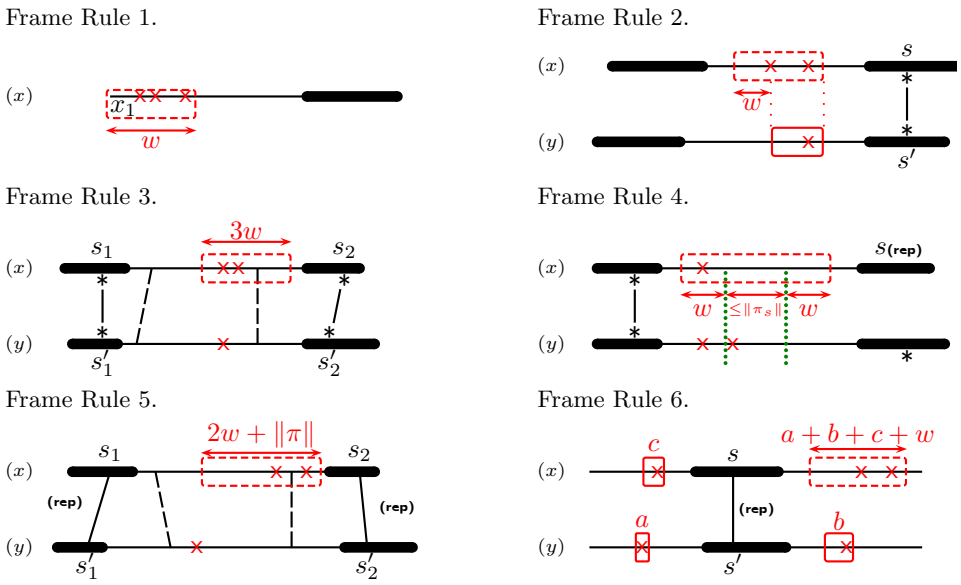

Fig. 4. Frame Rules 1–6 of `frames`. Frames are drawn as red boxes, the frame created at each step is dashed. Possible breakpoint positions in $\mathcal{P}$ are shown as red crosses.

– For an unfixed solid piece s, G contains the edge $\{v_f, v_s\}$ if the first marker of f is the last marker of either s or s' or if the last marker of f is the first marker of either s or s'.

Note that the vertices v_s or l_s and r_s are only defined for pieces $s \in S_x$, but they represent both pieces s and s'. Observe furthermore that in case $V_F \neq \emptyset$, there are fragile pieces in $\mathcal{C}$ that do not have a frame in Φ. Moreover, note that in this case the edge set of the piece graph is nonempty. Our aim will thus be to gradually apply the frame rules until the piece graph is edge-less. Each vertex is called *fragile*, *fixed* or *repetitive* depending on the nature of the piece it represents. Note that most vertices of the graph have degree at most 2, except for repetitive vertices which can have degree up to 4. Vertices with smaller degree correspond initially to the four pieces at the end of the sequences.

In order to deal seamlessly with pieces at the end of the input strings (where no fragile piece is adjacent on one side), we introduce "phantom frames" as follows. If s contains the first element of a string, i.e. $x[1]$ or $y[1]$, we say that s has the *phantom frame* $[x[0], x[1]]$ (resp. $[y[0], y[1]]$) to its left. Likewise, if s contains $x[n]$ or $y[n]$, it has the *phantom frame* $[x[n], x[n+1]]$ (resp. $[y[n], y[n+1]]$) to its right.

We now have collected the prerequisites to state the frame rules. A frame rule is an algorithm that receives as input a constraint $\mathcal{C}$ and a frame set Φ and updates both into a constraint $\mathcal{C}$' and a frame set Φ'. A frame rule is *correct* if following holds. First, if there is a size-k CSP $\mathcal{P}$ satisfying $\mathcal{C}$ and Φ, then there is also a size-k CSP $\mathcal{P}'$ satisfying $\mathcal{C}'$ and Φ'. Second, the longest undiscovered block in $\mathcal{P}'$ is at most as long as the longest undiscovered block in $\mathcal{P}$ (this additional restriction will be used to argue that the choice of β remains correct). Note that without loss of generality, we describe all rules for pieces in x but they apply to fragile pieces in x and y. Furthermore, if a rule works on a single fixed vertex in the piece graph, then we assume that this vertex is a left vertex l_s (by inverting the instance one can also deal with all right vertices). Finally, we state the additional frames of all rules by defining an interval which contains the window, in order to ensure that the frames are within the fragile pieces, we always intersect this interval with the considered fragile piece f. The first rule puts frames into fragile pieces at the end of the string.

Frame Rule 1. *If the piece graph contains a fragile degree-one vertex v_f, then f contains either $x[1]$ or $x[n]$. If f contains $x[1]$ add $f \cap [x[1], x[1] \triangleright w]$ to Φ, otherwise add $f \cap [x[n] \triangleleft w, x[n]]$ to Φ.*

Proof (of the correctness of Frame Rule 1). Fragile pieces of x that do not contain the first or the last marker of x are preceded and followed by a solid piece (since the splitting is alternating) and thus the corresponding vertex in the piece graph has degree two. Hence, a fragile piece in x corresponding to a degree-one vertex of the piece graph contains either the first or the last marker of x. Assume without loss of generality that f contains $x[1]$. The leftmost block of $\mathcal{P}$ in x is necessarily a short block since it is contained in the fragile piece f. Hence, marker $x[1]$ belongs to the first short block of $\mathcal{P}$ and it is next to a breakpoint of $\mathcal{P}$. Since the window (which contains all breakpoints in f) has length at most w, it is contained in the created frame $[x[1], x[1] \triangleright w]$. □

Frame Rule 2. *If the piece graph contains a degree-one vertex l_s with neighbor v_f such that f is next to s and s' does not contain $y[1]$, then: let $[s'^* \triangleleft u, s'^* \triangleleft v]$ be the (possibly phantom) frame to the left of s' in y; add the frame $f \cap [s^* \triangleleft (u + w - 1)), s^* \triangleleft v]$ to Φ.*

Proof (of the correctness of Frame Rule 2). Consider first the case where $[s'^* \triangleleft u, s'^* \triangleleft v]$ is a phantom frame: in this case, $s'^* \triangleleft v$ is $y[1]$ and $u = v + 1$. Hence, $y[1]$ is the first element of the block containing s'. Since $y[1]$ and $s^* \triangleleft v$ are equidistant from s, $s^* \triangleleft v$ is the first element of the block containing s and the last element of the window in f. Since the window has length at most $w - 1$, it is contained in the frame $[s^* \triangleleft (v + w), s^* \triangleleft v] = [s^* \triangleleft (u + w - 1)), s^* \triangleleft v]$.

Consider now the (regular) case where s' has a fragile piece g to its left. By the frame definition, all breakpoints of a satisfying CSP $\mathcal{P}$ that are in g are within $[s'^* \triangleleft u, s'^* \triangleleft v]$. Hence, $s'^* \triangleleft v$ is in the same block as s'. Consequently, the right limit of the window in f is to the left of $s^* \triangleleft v$ in f. Similarly, $s'^* \triangleleft u$ is in a different block than s' and thus there is a breakpoint to the right of $s^* \triangleleft u$ in f. All other breakpoints in f can have distance at most w from this breakpoint. Hence, all breakpoints in f are contained in the created frame $[s^* \triangleleft (u + w - 1), s^* \triangleleft v]$. □

The above rules are relatively straightforward inferences of frame positions that can be made because the piece graph has degree-one vertices. We now show some more intricate rules that deal with the remaining cases. In particular, we show how to deal with cycles in the piece graph. We first consider cycles without repetitive solid pieces. Note that the following rule performs a branching. We thus extend the correctness notion to hold if there is at least one branch in which the created constraint and frame set can be satisfied.

Frame Rule 3. *If the piece graph contains a simple cycle without repetitive vertices, then create one branch for each edge $\{v_f, u_s\}$ of this cycle. In each branch, add to Φ the frame*

- $f \cap [\mathrm{l}_{\mathrm{ext}}(s) \triangleleft w, \mathrm{l}_{\mathrm{ext}}(s) \triangleright (2w)]$ *if $u_s = l_s$ for some solid piece s, or*
- $f \cap [\mathrm{r}_{\mathrm{ext}}(s) \triangleleft 2w, \mathrm{r}_{\mathrm{ext}}(s) \triangleright (w)]$ *to f if $u_s = r_s$ for some solid piece s.*

The following is a straightforward property of constraints and satisfying solutions and used for showing the correctness of Frame Rule 3.

Proposition 3. *Let s be a fixed solid piece in a constraint $\mathcal{C}$. If markers a and a' are equidistant from s, then for any integer i, $a \triangleright i$ and $a' \triangleright i$ are equidistant from s. Moreover, given a CSP $\mathcal{P}$ satisfying $\mathcal{C}$, the first markers (the last markers) of the blocks of $\mathcal{P}$ containing s and s' are equidistant from s.*

Proof. The first part is directly obtained by definition:

$$\overline{s^*(a \triangleright i)} = \overline{s^* a} + i = \overline{s'^* a'} + i = \overline{s'^*(a' \triangleright i)}.$$

For the second part, simply note that if s^* is at position j in the block containing s, then s'^* is also at position j in s'. Hence, the first markers (and thus also the last markers) of both blocks are equidistant from s. □

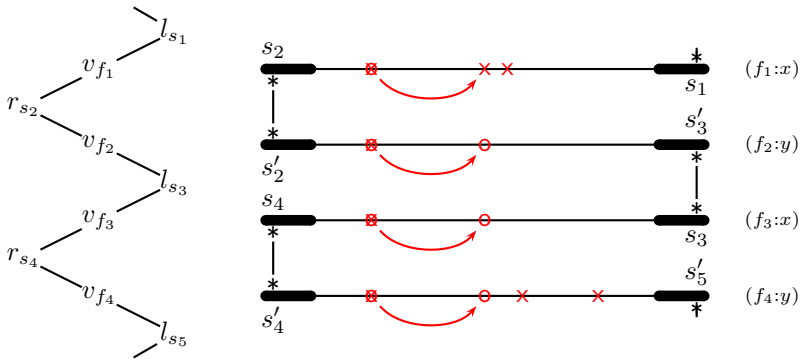

Fig. 5. Illustration for the first part of the correctness proof of Frame Rule 3. If two fragile windows f_i, f_j with different parity have several breakpoints (here, $i = 1$ and $j = 4$), then we can shift the position of the leftmost breakpoint in each fragile piece of the path to reduce the length of short blocks. The modifications (breakpoints added or deleted) are shown as red circles.

Proof (of the correctness of Frame Rule 3). Let $\mathfrak{P}$ be the set of CSPs that satisfy the constraint $\mathcal{C}$ and frame set Φ and additionally have a minimum total length of short blocks. We show that there is a $\mathcal{P} \in \mathfrak{P}$ which has all breakpoints in $[\mathrm{l}_{\mathrm{ext}}(s) \triangleleft w, \mathrm{l}_{\mathrm{ext}}(s) \triangleright (2w)]$ for some vertex l_s of the cycle, thus showing correctness of the rule.

Since the piece graph $G[\mathcal{C}, \Phi]$ is bipartite with partition V_S and V_F, the cycle alternates between vertices of V_S and V_F. Moreover, all cycle vertices from V_S are fixed, and alternate between left and right vertices (each fragile vertex of the cycle is adjacent to a left vertex and to a right vertex). Hence there exist solid pieces $s_1, s_2, \ldots, s_\ell$ and fragile pieces $f_1, f_2, \ldots, f_\ell$ such that the cycle is $(l_{s_1}, v_{f_1}, r_{s_2}, v_{f_2}, \ldots, l_{s_{\ell-1}}, v_{f_{\ell-1}}, r_{s_\ell}, v_{f_\ell})$. For simplicity, we consider indices only modulo ℓ (that is, $s_{\ell+1} = s_1$, $f_0 = f_\ell$, etc.), and we assume that fragile pieces with odd indices are in x and those with even indices are in y. Consider a CSP $\mathcal{P} \in \mathfrak{P}$ such that there is no l_s whose window is contained in $[\mathrm{l}_{\mathrm{ext}}(s) \triangleleft w, \mathrm{l}_{\mathrm{ext}}(s) \triangleright (2w)]$. We transform this CSP into one that fulfills this property. We first prove that in $\mathcal{P}$ either all fragile pieces with odd or all fragile pieces with even indices contain only one breakpoint. Assume towards a contradiction, that there exist integers $i < j$ of different parity such that f_i and f_j both have windows with at least two breakpoints and for each h with $i < h < j$, f_h contains only one breakpoint. Assume without loss of generality that i is odd and j is even. Hence, f_i is in x to the right of s_{i+1} and f_j is in y to the right of s_j.

For all h, $i \leq h \leq j$, let a_h be the leftmost marker of the window in f_h, and $b_h = a_h \triangleright 1$. For odd h, a_h and a_{h+1} are the rightmost markers of the blocks containing s_{h+1} and s'_{h+1} and thus equidistant from s_{h+1}. For even $h < j$, b_h and b_{h+1} are the left endpoints of the blocks containing s_{h+1} and s'_{h+1}, so they are equidistant from s_{h+1}. By Proposition 3, for all $i \leq h < j$, $[a_h, b_h]$ and $[a_{h+1}, b_{h+1}]$ are equidistant from s_{h+1}. By definition of a_h, the window in each f_h is contained in $[a_h, a_h \triangleright w]$. If one of these intervals is not contained in the maximum extension of an adjacent solid piece, say $[a_h, a_h \triangleright w]$ is not contained in the maximum extension of s_{h+1}, then $\mathrm{l}_{\mathrm{ext}}(s_{h+1})$ is contained in $[a_h, a_h \triangleright w]$. Hence, the window is contained in $[\mathrm{l}_{\mathrm{ext}}(s_{h+1}) \triangleleft w, \mathrm{l}_{\mathrm{ext}}(s_{h+1}) \triangleright w]$, contradicting our assumption on $\mathcal{P}$. In the following, we thus assume that all intervals $[a_h, a_h \triangleright w]$ are contained in the maximum extension of adjacent solid pieces, which by Proposition 1 implies that they all have the same content. In particular, this implies $[a_i, a_i \triangleright w] \equiv [a_j, a_j \triangleright w]$.

We now describe a modification of $\mathcal{P}$ that results in a new CSP which is not larger than $\mathcal{P}$, also satisfies the constraint and frame set but has smaller total length of short blocks; the modification is illustrated in Figure 5. Let $u + 1$ and $v + 1$ be the lengths of the leftmost short blocks in f_i and f_j respectively (assume without loss of generality that $u \leq v$). These two short blocks are thus $[b_i, b_i \triangleright u]$ and $[b_j, b_j \triangleright v]$, and they are matched in $\mathcal{P}$ to other short blocks $[b'_i, b'_i \triangleright u]$ and $[b'_j, b'_j \triangleright v]$. Note that since f_i is odd and f_j is even, $[b_i, b_i \triangleright u]$ is in a different string than

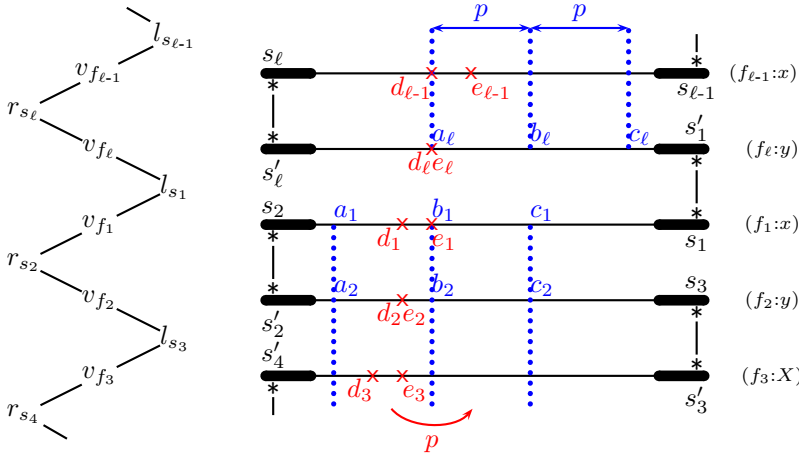

Fig. 6. Illustration for the correctness proof of the second part of Frame Rule 3. Given a cycle with total length of short blocks $p \geq 0$ we construct intervals $[a_h, b_h]$ and $[b_h, c_h]$ as shown (delimited by blue dotted lines). All the breakpoints in intervals $[a_h, b_h]$ can be shifted to the corresponding $[b_h, c_h]$.

$[b_j, b_j \triangleright v]$. To create the new solution $\mathcal{P}$' from $\mathcal{P}$ apply the following modifications. First, cut out $u+1$ markers from the left of $[b'_j, b'_j \triangleright v]$ (recall that $u \leq v$) which gives two new blocks $[b'_j, b'_j \triangleright u]$ and $[b'_j \triangleright (u+1), b'_j \triangleright v]$ if $u < v$ and leaves the block $[b'_j, b'_j \triangleright v]$ unmodified if $u = v$. Now, match block $[b'_i, b'_i \triangleright u]$ to $[b'_j, b'_j \triangleright u]$ (recall that these blocks are in different strings). Now, shift the breakpoints of the fragile pieces of the cycle as follows. For every odd h, $i < h < j$, cut out $u+1$ markers from the left of s'_h and s_h. And for every even h, $i < h \leq j$, add $u+1$ markers to the right of the blocks containing s_h and s'_h. Finally, in case $u < v$, match the shortened block $b_j \triangleright (u+1), b_j \triangleright v$ to the block $b'_j \triangleright (u+1), b'_j \triangleright v$ created in the first step. Note that by the discussion above, the pieces added to s_h and s'_h for even h have the same content. Hence, all matched blocks have equal content. Furthermore, since the block $[b_i, b_i \triangleright u]$ is now unmatched, its markers are free to be added to s_{i+1}.

This new solution has at most as many blocks as $\mathcal{P}$: we have created at most one new breakpoint in $[b'_j, b'_j \triangleright v]$ and removed a breakpoint in f_i by adding exactly $u+1$ markers to the right of s_{i+1}. For all other fragile pieces f_h, the breakpoint has "only" been shifted to the right. Furthermore, $\mathcal{P}'$ satisfies the same constraint $\mathcal{C}$ as $\mathcal{P}$: the matching only changed between short blocks which are not constrained. Moreover, the fragile pieces for which the breakpoints have been modified are either frameless (if they are on the cycle) or the modification adds a breakpoint that is between two breakpoints (in the modification of $[b'_j, b'_j \triangleright v]$) However,the total length of the short blocks has been reduced by $2(u+1)$, which contradicts the choice of $\mathcal{P}$. We now know that in $\mathcal{P}$ the short blocks of the cycle are either all in x or all in y. In the following, we assume they are all in x, that is, in fragile pieces f_j with odd j. We now consider the following two cases: either there is no short block, even in x, or there is at least one.

First consider the case that there is no short block in the cycle, that is, all the windows contain only one breakpoint $[a_h, b_h]$. If all markers b_h are within the maximum extensions of both adjacent solid pieces, we create a new solution $\mathcal{P}'$ from $\mathcal{P}$ as follows: for every odd h, cut out b_h and b_{h-1} from the left end of the blocks containing s_h and s'_h, and for every even h, add b_h and b_{h-1} to the right end of the blocks containing s'_h and s_h. The solution $\mathcal{P}'$ satisfies the same constraints as $\mathcal{P}$, with the same total length of short blocks. Repeat this operation of shifting the breakpoints to the right until for some i (without loss of generality, assume i is even), b_i is to the right of $\mathrm{r}_{\mathrm{ext}}(s_i)$. Then, the rule is correct, since for some branch the edge (r_{s_i}, v_{f_i}) is selected and the frame $[\mathrm{r}_{\mathrm{ext}}(s_i) \triangleleft 2w, \mathrm{r}_{\mathrm{ext}}(s_i) \triangleright w]$ which contains the only breakpoint of $\mathcal{P}$ in f_i is added to Φ.

It remains to show the case where there is at least one short block in the fragile pieces of the cycle, that is, the total length p of the short blocks of the cycle is at least one. Note that

by the choice of w, $p < w$. We now show that the strings around the windows are periodic with period length p, so that we can again shift all the breakpoints of the fragile pieces to the right by steps of length p, until at least one of them has distance at most p from the end of a maximum extension.

We first introduce some notations (see Figure 6 for an illustration): for each h, let $[d_h, e_h]$ denote the window of f_h. Let $b_1 = e_1$, $a_1 = b_1 \triangleleft p$, $c_1 = b_1 \triangleright p$, and for each h, $2 \leq h \leq \ell$, let a_h, b_h, and c_h be the markers equidistant with a_{h-1}, b_{h-1}, and c_{h-1} from s_h.

We first show that for every h with $2 \leq h \leq \ell$, we have

$$\overline{e_h b_h} = \overline{d_{h-1} b_{h-1}} - 1. \tag{1}$$

For even values of h, d_{h-1} and d_h are equidistant from s_h, so $\overline{d_h b_h} = \overline{d_{h-1} b_{h-1}}$. Since f_h is in string y, it contains only one breakpoint, and thus $\overline{d_h e_h} = 1$ and Equation (1) follows. For odd values of h, we have $\overline{d_{h-1} e_{h-1}} = 1$, and e_{h-1} and e_h are equidistant from s_h, thus $\overline{e_h b_h} = \overline{e_{h-1} b_{h-1}}$, which also implies equation (1). Hence the distance between window endpoint e_h and the marker b_h increases, compared to the distance of e_{h-1} and b_{h-1} by the length of the short blocks contained in the window of f_{h-1}. This has two implications: first, in f_ℓ, we have $\overline{e_\ell b_\ell} = p$ and thus $e_\ell = a_\ell$ (by definition a_1 has distance p from b_1, and this distance is conserved through the cycle). Second, for every j, the short blocks in f_j are contained in $[a_j, b_j]$, and the window is contained in $[a_j \triangleleft 1, b_j]$.

First, consider the case where each interval $[a_h, c_h]$ is contained in the maximal extensions of both adjacent blocks. Thus, with Proposition 1, we have $[a_h, b_h] \equiv [a_1, b_1]$ and $[b_h, c_h] \equiv [b_1, c_1]$ for all h. We can now "close" the cycle: since e_ℓ and e_1 are the left endpoints of the blocks containing s'_1 and s_1, they are aligned wrt. s_1. Moreover, $e_\ell = a_\ell$ and $e_1 = b_1$, so a_ℓ and b_1 are equidistant from s_1, which implies that $[a_\ell, b_\ell] \equiv [b_1, c_1]$. This now implies that, for all h, $[a_h, b_h] \equiv [b_h, c_h]$. We now create a solution $\mathcal{P}'$ from $\mathcal{P}$ as follows: for odd values of h, cut out the p leftmost markers from each block containing s_h or s'_h. For even values of h, add p markers to the right of blocks containing s_h or s'_h for even values of h. Match every short block that was matched to some $[u, v]$ in some f_h to $[u \triangleright p, v \triangleright p]$ instead. The solution $\mathcal{P}'$ is again a CSP satisfying the same constraints, with the same total length of short blocks but with all the breakpoints in the cycle shifted to the right by p positions. Repeat this operation until for some h the interval $[a_h, c_h]$ is no longer contained in the maximal extension of the block to its right. Then, $[a_h, c_h]$ contains $\mathrm{l}_{\mathrm{ext}}(s_h)$, and interval $[a_h \triangleleft 1, b_h]$ is contained in $[\mathrm{l}_{\mathrm{ext}}(s_h) \triangleleft 2w, \mathrm{l}_{\mathrm{ext}}(s_h) \triangleright w]$. As argued above, the rule is correct if such a $\mathcal{P} \in \mathfrak{P}$ exists. Note that the modifications made in the proof do not increase the length of any short block. Hence, the second requirement for correctness is also satisfied. □

The rules presented so far deal with fixed solid pieces. In fact, if all solid pieces are fixed, then these rules suffice to obtain frames in all fragile pieces. With the following three rules, we thus deal with the presence of repetitive solid pieces.

5 Frame Rules for Repetitive Pieces

In the rules, we have to deal with cycles in the piece graph that contain some repetitive vertices. We introduce the following concepts in order to analyze the structure of paths between repetitive vertices that contain fixed solid vertices. A rep−rep path $(v_s, v_{f_1}, u_1, v_{f_2}, u_2, \ldots, u_{\ell-1}, v_{f_\ell}, v_t)$ is a simple path of the piece graph such that the two endpoints v_s and v_t are repetitive vertices, and each u_i is a fixed solid vertex. Given a rep−rep path joining repetitive vertices v_s, v_t and going through fragile vertices $v_{f_1}, v_{f_2}, \ldots, v_{f_\ell}$, we define the *strip* of the path (see Figure 7) as a set of intervals $\{I_{f_1}, I_{f_2}, \ldots, I_{f_\ell}\}$ such that:

1. Consecutive intervals $I_{f_i}, I_{f_{i+1}}$ are equidistant from the solid piece represented by u_i.
2. Each interval I_{f_i} is contained in the maximum extensions of the two solid pieces next to f_i.

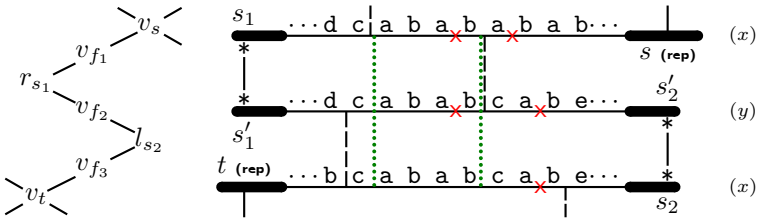

Fig. 7. Example of a rep−rep path joining repetitive vertices v_s and v_t (with respective periods `ab` and `ababc`), and going through three fragile vertices and their adjacent fixed vertices. The strip of each fragile piece is delimited by the green dotted lines.

3. The length of I_{f_1} is maximal under Conditions 1 and 2.

Proposition 4. *All the strips in a rep−rep path have the same length and content. Each interval of the strip is contained in its respective fragile piece. Moreover, the strip of a rep−rep path is uniquely defined and computable in polynomial time.*

Proof (of Proposition 4). The fact that strips have the same length and content is a direct consequence of Proposition 1, which can be applied according to Conditions 1 and 2. Each strip must be contained in its fragile piece since it is in the intersection of the maximum extensions of the two adjacent solid pieces.

The second part of the claim can be seen by considering the following algorithm to compute the strip. First, check whether the strip is nonempty. That is, try the following for each marker a_1 in f_1. Compute the marker a_2 in f_2 that is equidistant with a_1 from u_1. Then, compute the marker a_3 in f_3 that is equidistant with a_2 from u_2, and so on. If all a_i's are in the maximum extensions of both solid pieces next to f_i, then the strip is nonempty. Otherwise, the length of I_{f_1} is zero. Now, assume the case that there was one a_1 for which the above procedure is successful, that is, I_{f_1} contains one or more markers. Then, set $I_{f_i} := \{a_i\}$ for each i. Now try to simultaneously expand all I_{f_i}'s. That is, check whether one can add the marker to the left of each I_{f_i} without violating Condition 2 of the strip definition. If this is the case, then add these markers to the I_{f_i}'s. If this is not the case, then continue by adding markers to the right until this is also not possible anymore. The resulting set of I_{f_i}'s is the strip of the rep−rep-path. □

Proposition 5. *Let $\mathcal{P}$ be any solution satisfying constraint $\mathcal{C}$ such that the total length of all windows in $\mathcal{P}$ is p. In each fragile piece f of a rep−rep path of $\mathcal{C}$, writing $I_f = [c, d]$, the window of f is contained in $[c \triangleleft p, d \triangleright p]$.*

Proof (of Proposition 5). We first introduce some notations: let $f_1, f_2, \ldots, f_\ell$ be the fragile pieces of the path, and, for every $1 \leq j \leq \ell$, let $[a_j, b_j]$ denote the window of f_j, $I_{f_j} = [c_j, d_j]$, $\alpha_j = \overline{d_j a_j}$ and $\beta_j = \overline{d_j b_j}$.

Hence we aim at showing that for all j, $\beta_j \leq p$, that is, b_j is either to the left or at at most p markers to the right of d_j. The proof for the left bound, that is, to show that a_j is at most p markers to the left of c_j is symmetrical.

By maximality of the strip length (Condition 3), the intervals of the strip cannot be extended to the right. Condition 2 is the one constraining the strip length, hence there exists a fragile piece f_{j_0} in the path such that this condition is tight, that is, $d_{j_0} = \mathrm{r}_{\mathrm{ext}}(s)$, where s is the solid piece to the left of f_{j_0}. Hence, a_{j_0} is not to the right of d_{j_0}, and thus $\alpha_{j_0} = \overline{d_{j_0} a_{j_0}} = \overline{\mathrm{r}_{\mathrm{ext}}(s) a_{j_0}} \leq 0$.

Now for all j, $\beta_j - \alpha_j = \|[a_j, b_j]\| - 1$, that is, it is the length of the window contained in f_j minus one. Consequently, $\beta_{j_0} < \|[a_{j_0}, b_{j_0}]\|$. Moreover, for every $1 \leq j < \ell$, either the first markers of the window of f_j and f_{j+1} are matched and thus equidistant to the piece represented by u_i or the last markers of the window of f_j and f_{j+1} are matched and thus equidistant to u_i. Hence, either $\alpha_j = \alpha_{j+1}$ or $\beta_j = \beta_{j+1}$. In the first case, β_{j+1} increases, compared to β_j, by at

most $\|[a_{j+1}, b_{j+1}]\| - 1$. Hence, $\beta_j \leq \beta_{j_0} + p$ for all $j \geq j_0$. By symmetry, the same holds for all $j \leq j_0$. □

The following rule serves as a "preparation" of our main rule that deals with cycles containing repetitive vertices. It will ensure that if there is a cycle containing repetitive vertices, then these repetitive vertices will have the same period.

Frame Rule 4. *If the piece graph contains a rep−rep path between repetitive vertices v_s and v_t with strip $\{I_{f_1}, \ldots, I_{f_\ell}\}$ such that the strip $I_f = [u, v]$ in f is shorter than the period π_s of s, then add the frame $f \cap [u \triangleleft w, v \triangleright w]$ to f.*

Proof (of the correctness of Frame Rule 4). By definition, w is at least the total length of the windows of $\mathcal{P}$. By Proposition 5, the endpoints of the window of f thus have distance at most w from I_f. □

Frame Rule 5. *If Frame Rule 4 does not apply and the piece graph contains a simple cycle with repetitive vertices, then do the following. Let $\|\pi\|$ be the length of the period of any repetitive solid piece in the cycle. Then, create one branch for each edge $\{v_f, u_s\}$ of the cycle where u_s is a solid vertex for the solid piece s. In each branch, add to Φ the frame*

- *$f \cap [\mathrm{r}_{\mathrm{ext}}(s) \triangleleft (\|\pi\| + w), \mathrm{r}_{\mathrm{ext}}(s) \triangleright w]$ if f is to the right of s, or*
- *$f \cap [\mathrm{l}_{\mathrm{ext}}(s) \triangleleft w, \mathrm{l}_{\mathrm{ext}}(s) \triangleright (\|\pi\| + w)]$ if f is to the left of s.*

Proof (of the correctness of Frame Rule 5). First, all repetitive pieces of the path have the same period. Indeed, consider any two consecutive repetitive pieces s and t of the cycle: they are linked by a rep−rep path, in which we compute the strips. All strips in this path have equal length S and also equal content (Proposition 4). Hence, the maximal extensions of repetitive pieces s and t have a common substring of length S. Since Frame Rule 4 does not apply, we have $\|\pi_s\| \leq S$ and $\|\pi_t\| \leq S$. Thus, the maximum extensions of s and t contain a common substring longer than their respective periods. Consequently, their periods are equal, and thus all repetitive pieces of the cycle have the same period π.

Let $s_1, s_2, \ldots, s_\ell$ denote the repetitive pieces crossed successively by the cycle (again, we write $s_{\ell+1} = s_1$). For each i, $1 \leq i \leq \ell$, let $x_i^{\triangleleft}$, $x_i^{\triangleright}$, $y_i^{\triangleleft}$, $y_i^{\triangleright}$ be the fragile pieces to the left and right of s_i in x and s'_i in y, respectively. For each rep−rep path of the cycle from s_i to s_{i+1}, we say the path is *positive* if the first vertex after s_i is $v_{x_i^{\triangleleft}}$ or $v_{y_i^{\triangleright}}$, and *negative* otherwise. In positive rep−rep paths, fragile pieces in x are crossed from right to left (that is, the solid piece to the right of the fragile piece is "seen" before the solid piece to its left), and fragile pieces in y are crossed from left to right. Thus a positive path enters s_{i+1} via either $v_{x_{i+1}^{\triangleright}}$ or $v_{y_{i+1}^{\triangleleft}}$, and likewise a negative path enters s_{i+1} via either $v_{x_{i+1}^{\triangleleft}}$ or $v_{y_{i+1}^{\triangleright}}$.

First, consider the case that all windows are contained within the strip and that both endpoints of the piece have distance at least $\|\pi\|$ to the borders of the strip. We show that in this case, we can shift all breakpoints in positive paths to the right by step $\|\pi\|$ positions and all breakpoints in negative paths to the left by $\|\pi\|$ positions. This is done as follows:

- For each fixed vertex l_s in a positive path, cut out $\|\pi\|$ markers from the left of the blocks containing s and s'.
- For each fixed vertex r_s in a positive path, add $\|\pi\|$ markers from the right of the blocks containing s and s'.
- For each fixed vertex l_s in a negative path, add $\|\pi\|$ markers to the left of the blocks containing s and s'.
- For each fixed vertex r_s in a negative path, cut out $\|\pi\|$ markers from the right of the blocks containing s and s'.
- Replace each short block $[a, b]$ in a fragile piece of a positive path by $[a \triangleright \|\pi\|, b \triangleright \|\pi\|]$.
- Replace each short block $[a, b]$, in a fragile piece of a negative path by $[a \triangleleft \|\pi\|, b \triangleleft \|\pi\|]$.

– For a repetitive vertex v_{s_i} such that the paths before and after v_{s_i} enter and leave v_{s_i} via the same side (either $x_i^{\triangleleft}$ and $y_i^{\triangleleft}$, or $x_i^{\triangleright}$ and $y_i^{\triangleright}$) either both paths are positive or both paths are negative. Apply the same operation as if the piece was fixed:
  - If the path enters v_{s_i} via $x_i^{\triangleleft}$ and leaves via $y_i^{\triangleleft}$, then cut out the $\|\pi\|$ leftmost markers of s and s' if the path is positive or add the $\|\pi\|$ markers to the left of s and s' if the path is negative.
  - If the path enters v_{s_i} via $x_i^{\triangleright}$ and leaves via $y_i^{\triangleright}$, then cut out the $\|\pi\|$ rightmost markers of s and s' if the path is negative or add the $\|\pi\|$ markers to the right of s and s' if the path is positive.
– For a repetitive vertex v_{s_i} such that the paths enter and leave the vertex via the same string (either $x_i^{\triangleleft}$ and $x_i^{\triangleright}$, or $y_i^{\triangleleft}$ and $y_i^{\triangleright}$) it holds that the paths have the same orientation. Apply a similar operation as for a short block (assume without loss of generality that the path enters and leaves via x):
  - If v_{s_i} is between two positive paths then replace the block $[a, b]$ of x containing s_i by $[a \triangleright \|\pi\|, b \triangleright \|\pi\|]$.
  - If v_{s_i} is between two negative paths then replace the block $[a, b]$ of x containing s_i by $[a \triangleleft \|\pi\|, b \triangleleft \|\pi\|]$.
– For all other repetitive vertices, the paths enter from one string and leave via the other string and enter from one side and leave via the other side. Then the paths have opposite orientations; assume without loss of generality that the entering path is positive and the outgoing path is negative. Let $[a, b]$ denote the block in x containing s_i, and let $[a', b']$ denote the block in y containing s'_i.
  - If the cycle enters from $y_i^{\triangleleft}$ and leaves via $x_i^{\triangleright}$, then replace $[a, b]$ by $[a, b \triangleleft \|\pi\|]$ and $[a', b']$ by $[a' \triangleright \|\pi\|, b']$ ($\|\pi\|$ markers are cut out of both blocks).
  - If the cycle enters from $x_i^{\triangleright}$ and leaves via $y_i^{\triangleleft}$, then replace $[a, b]$ by $[a, b \triangleright \|\pi\|]$ and $[a', b']$ by $[a' \triangleleft \|\pi\|, b']$ ($\|\pi\|$ markers are added to both blocks).

Thus, all the breakpoints in fragile pieces have been shifted to the right (in positive paths) or to the left (in negative paths) by a period length $\|\pi\|$. Hence, this modification still gives a partition of both strings. This partition has the same size as the original one. Furthermore, it is also a common string partition which can be seen as follows. The set of strings represented by the short blocks of x and y remains exactly the same since they were shifted by the period length. Hence, there is matching for the short blocks such that each short block is matched to one representing the same string. For the long blocks, the old matching remains a valid matching: The blocks containing fixed solid pieces have both been modified on the same side. Thus, they are either shortened by $\|\pi\|$ markers; in this case, the matched blocks clearly represent equivalent strings. Or $\|\pi\|$ markers have been added on one side. In this case, the matched strings are also equivalent, since the windows have distance at least $\|\pi\|$ to the borders of the strip. The blocks containing repetitive pieces have either been moved by $\|\pi\|$ positions, shortened by $\|\pi\|$ markers on the same side, $\|\pi\|$ markers on the same side have been added, or they have been shortened or extended on different sides. In the first three cases, the strings represented by the new blocks remain equivalent for the same reasons as for the blocks containing fixed solid pieces. It remains to show the case in which blocks have been modified on different sides.

First, consider the case in which $[a, b]$ is replaced by $[a, b \triangleleft \|\pi\|]$ and $[a', b']$ by $[a' \triangleright \|\pi\|, b']$. Since the blocks are periodic with period length $\|\pi\|$ we have $[a' \triangleright \|\pi\|, b'] \equiv [a', b' \triangleleft \|\pi\|]$. In the old solution, this subinterval of $[a', b']$ was matched with $[a, b \triangleleft \|\pi\|]$, and thus $[a' \triangleright \|\pi\|, b'] \equiv [a', b' \triangleleft \|\pi\|] \equiv [a, b \triangleleft \|\pi\|]$.

Now consider the case in which $[a, b]$ is replaced by $[a, b \triangleright \|\pi\|]$ and $[a', b']$ by $[a' \triangleleft \|\pi\|, b']$. Since the blocks are periodic with period length $\|\pi\|$ we have $[a', b'] \equiv [a' \triangleleft \|\pi\|, b' \triangleleft \|\pi\|]$. Since $[a, b] \equiv [a', b']$ this implies that the first $\|[a, b]\|$ markers of the new blocks are equivalent.

Also because of the periodicity, we have $[b, b \triangleright \|\pi\|] \equiv [b \triangleleft \|\pi\|, b]$. Since $[b \triangleleft \|\pi\|, b] \equiv [b' \triangleleft \|\pi\|, b']$, this implies that also the last $\|\pi\|$ markers of the new blocks are equivalent.

Altogether, the modification gives a CSP of the same size, in which the distance between the window endpoints and the strip endpoints has decreased. The above operation can be repeated until at least one breakpoint is at distance less than $\|\pi\|$ from the border of a strip. In this case, all breakpoints of the corresponding path are at distance at most $w + \|\pi\|$ from the border of their corresponding strip (an argument similar to the proof of Proposition 5 applies). In some fragile piece f, the border of I_f coincides with the maximum extension of an adjacent solid piece s, thus, in f, the window is contained in either $[\mathrm{l}_{\mathrm{ext}}(s) \triangleleft (\|\pi\| + w), \mathrm{l}_{\mathrm{ext}}(s) \triangleright w]$ or $[\mathrm{r}_{\mathrm{ext}}(s) \triangleleft w, \mathrm{r}_{\mathrm{ext}}(s) \triangleright (\|\pi\| + w)]$. Since in one of the considered branches, the rule adds the frame to this piece s and to the correct side of the strip interval it is correct. Note that the modifications made in the proof do not increase the length of any short block. Hence, the second requirement for correctness is also satisfied. □

The final case that needs to be considered is the one in which the piece graph is acyclic but none of the other rules applies. Then, the piece graph contains a repetitive degree-one vertex.

Frame Rule 6. *If the piece graph contains an edge $\{v_s, v_f\}$ such that v_s is repetitive and has degree one, then assume without loss of generality that f is to the right of s in x, and do the following. Let $[a_l, a_r]$, $[b_l, b_r]$, and $[c_l, c_l]$ be the (possibly phantom) frames such that $[a_l, a_r]$ is to the left of s' in y, that $[b_l, b_r]$ is to the right of s' in y, and that $[c_l, c_r]$ is to the left of s in x. Add the frame $f \cap [f_l, f_r]$ to f, where $f_l := c_l \triangleright (\overline{a_r b_l} + 1)$ and $f_r := c_r \triangleright (\overline{a_l b_r} + w - 2)$.*

Proof (of the correctness of Frame Rule 6). The window to the left and right of s' in y are contained in $[a_l, a_r]$ and $[b_l, b_r]$ respectively, and the window to the left of s in x is contained in $[c_l, c_r]$. Consider the blocks containing s and s', and let ℓ be their length. The two endpoints of the block containing s' are in $[a_l \triangleright 1, a_r]$ and $[b_l, b_r \triangleleft 1]$. Hence $\ell \geq \overline{a_r b_l}$ and $\ell \leq \overline{(a_l \triangleright 1)(b_r \triangleleft 1)} = \overline{a_l b_r} - 2$.

The leftmost marker of the block containing s is contained in $[c_l \triangleright 1, c_r]$. Thus, the rightmost marker (the one in f) is necessarily in $[c_l \triangleright (\ell + 1), c_r \triangleright (\ell)]$ which, by the above upper and lower bounds on ℓ, is contained in $[c_l \triangleright (\overline{a_r b_l} + 1), c_r \triangleright (\overline{a_l b_r} - 2)]$. This marker is the leftmost marker of the window of f which has length at most w. Hence the frame $[c_l \triangleright (\overline{a_r b_l} + 1), c_r \triangleright (\overline{a_l b_r} + w - 2)]$ contains the window of f. The rule is still correct if s or s' corresponds to the end of a string, since the phantom frames contain the leftmost or rightmost marker of the blocks containing s or s'. □

After exhaustively applying the frame rules, parts of fragile pieces that are outside of frames do not contain a breakpoint. Hence, we perform the following rule which shrinks fragile pieces such that they fit their frame; at the same time, the solid pieces are extended accordingly.

Fitting Rule 1 *If there is a fragile piece $f = [a, b]$ with frame $[c, d]$ such that $a \neq c$ or $b \neq d$, then add $[a, c]$ to the solid piece left of f, add $[d, b]$ to the solid piece right of f, and set $f := [c, d]$.*

Fitting Rule 1.

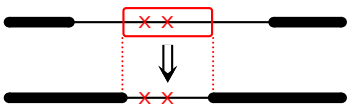

Fig. 8. An illustration of Fitting Rule 1 of `frames`

We now show two important properties of instances for which none of the frame rules applies. First, every fragile piece of these instances has a frame. Second, the frame lengths are upper-bounded by a function of k, β, and the longest period of any repetitive piece.

Lemma 5. *Let $\mathcal{C}$ be a constraint with frame set ϕ such that none of the Frame Rules 1–6 applies. Then, each fragile piece has a frame, and all frames have length at most $(6k^2w + 3kw + 3k\max\{w, \|\pi\|\})$, where π denotes the length of the longest period of all repetitive solid pieces.*

Proof (of Lemma 5). First, we show that every fragile piece has a frame. If the piece graph contains a cycle, then either Frame Rule 3, 4, or 5 applies. Otherwise, the piece graph is acyclic, and thus it either contains a degree-one vertex and one of the other Frame Rules applies, or all vertices have degree zero which means that all fragile pieces have frames.

Next, we show the upper bound on the frame length. Let L be the length of the longest frame created in this procedure, and let π be the longest period over all repetitive pieces. We show that

$$L \leq 6k^2w + 3kw + 3k\max\{w, \|\pi\|\} \tag{2}$$

Let h be the number of frames created before Frame Rule 6 is first applied, $1 \leq h \leq 2k$. Rules 1, 3, 4 and 5 produce frames of length at most $(\max\{w, \|\pi\|\}+2w)$. Since each application of Rule 2 increases the maximum frame length by w, all frames have length at most $(\max\{w, \|\pi\|\} + (h+1)w)$ before the first application of Frame Rule 6. Note that once Rule 6 is applied for the first time, only Rules 2 and 6 can be applied. We introduce the following notations. A solid vertex (fixed or repetitive) is *closed* if all its adjacent fragile pieces have frames, and *open* otherwise. The *weight* of an open vertex is the total length of the frames in the adjacent fragile pieces. Let W denote the sum of the weights of all open vertices.

Before the first application of Frame Rule 6, $W \leq 3k[\max\{w, \|\pi\|\} + (h+1)w]$ (for each solid piece s, the weight of either v_s or l_s and r_s together is at most the sum of the weights of three different frames). Afterwards, each time Rule 2 or 6 is applied, an open vertex with some weight u is closed, and a frame of length $u + w$ is created in a fragile piece f which is adjacent to at most one open vertex. Thus, the total weight of open vertices W is increased by at most $u + w - u = w$ with each application of Frame Rule 2 or 6. They are applied at most $2k - h$ times, hence W is at most

$$\begin{aligned} W &\leq 3k\left[\max\{w, \|\pi\|\} + (h+1)w\right] + (2k-h)w \\ &\leq 6k^2w + 3kw + 3k\max\{w, \|\pi\|\}. \end{aligned}$$

Since no frame of length more than W can be created, we have $L \leq W$, which proves the second part of the claim. □

The bound given by the lemma above still contains the maximum period length π which means that it is too large to be useful for the `split` procedure. However, the algorithm can now either find a repetitive piece which can be fixed with few options (see Lemma 6) or the maximum period length is not too long.

Lemma 6. *Let $\mathcal{C}$ be a constraint that contains a repetitive solid piece s with period π_s such that each fragile piece adjacent to s or s' has length at most $6(k^2+k)\|\pi_s\|$. Then, there are at most $12(k^2+k)$ feasible alignments, and any CSP satisfying $\mathcal{C}$ matches elements of s according to a feasible alignment.*

Proof (of Lemma 6). The alignment corresponding to any CSP satisfying $\mathcal{C}$ is necessarily feasible, since otherwise two distinct solid pieces would be contained in the same block.

Without loss of generality, let $\|s\| \geq \|s'\|$. Thus, in a satisfying CSP $\mathcal{P}$, either the leftmost marker of s is matched to a marker left of s' (or to the leftmost marker of s'), either the rightmost marker of s is matched to a marker right of s'. Consider the first case; by Condition 2 of satisfying CSPs, the leftmost marker of s is matched to some marker in the fragile piece to the left of s'. Note that since s and s' have a shortest period π_s, two different alignments are separated by a a multiple of $\|\pi_s\|$ markers. Hence, there are at most $6(k^2+k)$ different alignments

in which the leftmost marker of s is matched to some marker of the fragile piece to the left of s'. Similarly, there are at most $6(k^2+k)$ possible alignments in which the rightmost marker of s is matched to a marker of the fragile piece to the left of s' (for $\overline{s} = \overline{s'}$, it is possible that both left and right endpoints of s are matched to markers of the fragile pieces to the left and right of s'). Overall, the total number of alignments between s and s' thus is at most $12(k^2+k)$. □

By guessing the alignments of the long periods we have finally achieved the goal of `frames`: all frames are "short" enough to be split by `split`.

Lemma 4. *When* `frames` *terminates, every fragile piece has length at most* $12(k^2+k)k\beta$.

Proof (of Lemma 4). By Lemma 5, an instance in which no frame rule applies has frames of length at most $(6k^2w + 3kw + 3k\max\{w, \|\pi\|\})$ where π is the longest period among all repetitive pieces. In case $\|\pi\| \geq w$, then for each repetitive piece s with period π, there are at most $6(k^2+k) \cdot \|\pi\|$ possibilities to align s. Hence, at least one repetitive piece is fixed in the loop Lines 8–11, and `new-align` is set "True", which means that the outer loop in `frames` will be repeated. Otherwise, $\pi \leq w$ and thus $6k^2w + 3kw + 3k\|\pi\| < 6(k^2+k)w \leq 12(k^2+k)k\beta$. □

The correctness of `frames` is simply a consequence of the correctness of all single steps (always considering the correct branching in each branching step).

Lemma 2. *If there exists a size-k CSP* $\mathcal{P}$ *satisfying* $\mathcal{C}$ *at the beginning of* `frames` *such that longest undiscovered block is* β*-critical, then* `frames` *creates at least one branch such that the constraint in this branch is satisfied by a size-k CSP* $\mathcal{P}'$ *whose longest undiscovered block has length at most* $2\beta - 1$.

Proof (of Lemma 2). The correctness of all frame rules have already been proven. The correctness of Fitting Rule 1 is trivial. Finally, the correctness of Lines 8–11 follows simply from the fact that the alignment in one of the branches is the correct one (it considers all feasible alignments). Since the correctness definition of the frame rules demands that all undiscovered are at most as long as before adding the frame, also the size bound for the longest undiscovered block holds.

It thus remains to bound the running time of `frames`. In particular, we need to show that the number of branches is bounded by a function of k.

Lemma 3. *Overall, the calls to* `frames` *create* $(2k)^{4k^2} \cdot k^{O(k)}$ *branches; all other parts of the algorithm can be performed in* $\mathrm{poly}(n)$ *time.*

Proof (of Lemma 3). First, note that the outer repeat-until loop of `frames` is repeated at most $2k$ times over the course of *all* calls to `frames`: The procedure `frames` is called at most k times from the main method. Each additional time the repeat-until loop is repeated, there is a pair of repetitive solid pieces that becomes a pair of fixed solid pieces at Line 10 of the previous pass of the repeat-until loop. This can happen at most k times.

Second, note that the while loop of Lines 4–5 is iterated at most $2k$ times in each repetition of the other repeat-until loop of `frames`: each rule creates exactly one frame, and, by Observation 1 there are at most $2k-2$ fragile pieces.

Hence, there are at most $4k^2$ times in which one of the frame rules at Line 5 creates branches and at most k times in which branches are created at Line 10. The only frame rules that perform branchings are Frame Rules 3 and 5. In both cases, the rule branches into at most $2k$ cases, since each cycle has at most k solid vertices and thus at most $2k$ vertices edges in the cycle under consideration. Hence, the branchings performed by the frame rules increase the running time by a factor of $O((2k)^{4k^2})$. Each of the at most k branchings in Line 10 is among at most $(12k)^2 + k$ choices (Lemma 6). Hence, these branchings increase the running time by a factor of $O((12k)^{2k} \cdot k^k)$. Hence, the overall increase due to the branching is by a factor of $(2k)^{4k^2} \cdot k^{O(k)}$; all other steps can be performed in polynomial time.

6 Conclusion

An improvement of the so far very impractical running time is desirable; the bottleneck appears to that some the frame rules still have to branch. Furthermore, it would be interesting to see whether our result for MCSP can be extended to the "signed" MCSP [3, 5, 10] where each marker is annotated with a direction and one may reverse blocks before matching.

References

[1] V. Bafna and P. A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM J. Comput.*, 25(2): 272–289, 1996.

[2] V. Bafna and P. A. Pevzner. Sorting by transpositions. *SIAM J. Discrete Math.*, 11(2):224–240, 1998.

[3] X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 2(4):302–315, 2005.

[4] P. Damaschke. Minimum common string partition parameterized. In *Proc. 8th WABI*, volume 5251 of *LNCS*. Springer, 2008.

[5] G. Fertin, A. Labarre, I. Rusu, E. Tannier, and S. Vialette. *Combinatorics of Genome Rearrangements*. Computational Molecular Biology. MIT Press, 2009.

[6] B. Fu, H. Jiang, B. Yang, and B. Zhu. Exponential and polynomial time algorithms for the minimum common string partition problem. In *Proc. 5th COCOA*, volume 6831 of *LNCS*, pages 299–310. Springer, 2011.

[7] A. Goldstein, P. Kolman, and J. Zheng. Minimum common string partition problem: Hardness and approximations. *Electron. J. Comb.*, 12, 2005.

[8] H. Jiang, B. Zhu, D. Zhu, and H. Zhu. Minimum common string partition revisited. *J. Comb. Optim.*, 23: 519–527, 2012.

[9] D. Shapira and J. A. Storer. Edit distance with move operations. *J. Discrete Algorithms*, 5(2):380–392, 2007.

[10] K. M. Swenson, M. Marron, J. V. Earnest-DeYoung, and B. M. E. Moret. Approximating the true evolutionary distance between two genomes. *ACM J. Exp. Algorithmics*, 12, 2008.