Minimum Common String Partition Parameterized*

Peter Damaschke

Department of Computer Science and Engineering
Chalmers University, 41296 Göteborg, Sweden
ptr@cs.chalmers.se

Abstract. Minimum Common String Partition (MCSP) and related problems are of interest in, e.g., comparative genomics, DNA fingerprint assembly, and ortholog assignment. Given two strings with equal symbol content, the problem is to partition one string into k blocks, k as small as possible, and to permute them so as to obtain the other string. MCSP is NP-hard, and only approximation algorithms are known. Here we show that MCSP is fixed-parameter tractable in suitable parameters, so that practical instances can be efficiently solved to optimality.

1 Parameterization of MCSP

String z is called a *substring* of string x, if there exist strings u, v (possibly empty) such that $x = uzv$. The same substring z may appear at several positions in x. By a *segment* of x we mean an occurence of a substring at a specific position in x. A substring or segment may be empty, where an empty segment is defined by its location between two consecutive symbols. Length $|x|$ is the number of symbol occurences in x.

In the Minimum Common String Partition (MCSP) problem, two strings x, y of length n, that contain the same symbols equally many times, shall be partitioned each into k segments called *blocks*, so that the blocks in x and y constitute the same multiset of substrings. Cardinality k shall be minimized. An equivalent formulation is: Split x into k segments so that y equals the concatenation of some permutation of them. A position between two consecutive blocks is called a *break*, hence we have $k - 1$ breaks in each of x and y. MCSP is of interest in comparative genomics [4], to answer questions like: Is a DNA string possibly obtained by rearrangements of another DNA string? The problem is NP-hard even in very restricted cases [7], and several approximation algorithms and non-approximability results are known [7, 5, 8]. A related problem is Minimum Common Integer Partition (MCIP): Given two multisets of at most k integers, split these integers into a minimum number of integer summands so that the two resulting multisets are equal. Biological motivations include ortholog assignment and DNA fingerprint assembly. MCIP also appears as a subproblem in

* Supported by the Swedish Research Council (Vetenskapsrådet), grant no. 2007-6437, "Combinatorial inference algorithms – parameterization and clustering".

certain instances of MCSP. The complexity status is similar to that of MCSP [3, 12]. Due to lack of space we refer to [3, 4] for details about the biological background.

Since k and other input parameters are expected to be much smaller than n in biological applications, it is sensible to ask whether MCSP is fixed-parameter tractable (FPT). To our best knowledge, this issue has not been addressed before in the literature. A problem with input length n and one or more input parameters is in FPT, if it can be exactly solved in time polynomial in n, for any fixed parameter values, where the exponent in the polynomial bound must be constant, i.e., independent of the parameters. For an introduction to FPT see, e.g., [6]. Many problems in computational biology, among them sequence comparison problems, are NP-hard but in FPT, here we only refer to [2, 9].

Besides the number k of blocks we consider two more parameters for MCSP: We define the *distance ratio* $d = n/m$, where m is the minimum distance between breaks. A limited d means that we ignore (or do not expect) solutions where some blocks are very short fragments. We define the *repetition number* r of a string x as the maximum i so that $x = uv^iw$ holds for some strings u, v, w, where v is nonempty. Although repeats in genomes are not uncommon, the repetition number in biological sequences will often be small. (See the remark below.)

Contributions and organization of the paper:

The obvious question is whether MCSP is fixed-parameter tractable in k. While this remains open, we show fixed-parameter tractability in combined parameters k, r. For ease of presentation we step two paces back and derive first a weaker result in Section 2: an FPT algorithm for MCSP with combined parameters k, d, r. Actually, parameters d, r suffice, since obviously $k \leq d$ must be assumed. The algorithm has a simple structure: First we roughly guess the breaks and the matching of blocks "in parameterized time". Then we are left with a rather special linear system of equations that enables us to check whether a solution of this shape exists. Since we test all possible branches, the same approach can also be used to enumerate all solutions that respect the given parameters. (If alternative solutions exist, it is not clear which of them is the correct interpretation of data, hence one should consider them all. See [1] for another example.) The time is $O(n)$ for any fixed parameters. We remark that linear-time *approximation* algorithms for MCSP and the related reversal distance problem are known if other parameters are bounded, namely if every symbol may appear only constantly many times [10, 11]. This case is interesting, e.g., if the symbols represent genes. Since r cannot be larger than the number of appearances of every symbol, our result implies also that MCSP, with bounded number of appearances of every symbol, is in FPT.

We also state a simple explicit upper bound for the parametric part of the complexity bound, however it seems overly pessimistic. The restriction that aligned substrings be equal is so strong that the actual number of branches to consider is typically much smaller, although there might be malicious cases. In Section 3 we illustrate the issue and add some $O(n)$ time heuristics that dis-

card hopeless branches quickly. Simplicity of the scheme, and the observation that almost no branching takes place, should make the approach very practical.

In Section 4 we get rid of parameter d and replace it with $k \leq d$. That is, we also identify arbitrarily short blocks. The difficulty is to separate the close breaks, which requires different techniques and arguments. Our $O(n \log n)$ time algorithm (for any fixed k, r) applies the one from Section 2 in an iterative fashion on a decreasing scale. In Section 5 we point out that MCIP is polynomial for any fixed k (which is simple). Since MCIP is related to MCSP instances with high r, this gives rise to the conjecture that MCSP might be fixed-parameter tractable with k as the only parameter. Section 6 gives some outlook. Proofs are basically complete, but due to space limitations, some passages are only sketched and not much formalized, and no pictures to illustrate the algorithms could be included.

2 An FPT Algorithm for MCSP

A few definitions and basic facts around periodic strings will be needed. A nonempty string p is called a *period* of string z if $z = sp^it$, where $i > 0$ is an integer, s a suffix and t a prefix of p (both may be empty). Every string has a shortest period, uniquely determined up to cyclic shifts. If string z overlaps itself on $(1-\alpha)|z|$ symbols (i.e., prefix and suffix of z of that length are equal strings), where $0 < \alpha \leq 1$, then z has a period of length $\alpha|z|$. We will call certain segments of x or y *fragile* (*conserved*), when we decided to put (not to put) breaks there. Two segments without a symbol in between are *adjacent*.

Now we start describing our algorithm for parameters k, d, r. For some $c > d$ specified below, we split both x and y into c segments called *pieces*, of length roughly $n/c < n/d$. Since we consider only blocks of length at least n/d, a piece can contain at most one break. (Breaks exactly between two adjacent pieces are arbitrarily assigned to, e.g., the piece to the left.) We guess the $k-1$ pieces that contain breaks, these are our fragile segments. We branch on these choices. We also guess the *matching* between the resulting k blocks of x and y that shall be aligned, and branch on the possible matchings, i.e., permutations of blocks. Clearly, the number of branches is bounded by some function of k and c. Note that only the exact positions of breaks in the designated fragile segments are yet unknown in every branch.

If we choose $c \geq 2d$, at least one entire piece in every block is conserved. Now consider any pair of matched blocks s from x, and t from y. Let s' be the concatenation of all conserved pieces in s. Although the exact borders of s are yet undetermined, we know that s' is a subsegment of s. Let t^+ be the segment of y consisting of all pieces that are certainly in t or may contribute some symbols to t. (The latter ones are the delimiting fragile pieces. Informally, t^+ is the "maximal possible t".) Clearly, s' must be aligned to some segment of t^+. If s' occurs several times in t^+, we guess the alignment partner of s' in t^+, and branch on these choices. Assume that the start positions of two occurrences of s' in t^+ have a distance $\alpha|s'|$, $0 < \alpha \leq 1$. Then s' has a period of length $\alpha|s'|$,

which is repeated $1/\alpha$ times. It follows $\alpha \geq 1/r$, hence the number of branches is bounded by some function of c and r.

As $c \geq 2d$ was the only condition on c, we may choose $c = 2d$. Now the *total* number of branches (deciding on: pieces with breaks, matching of blocks, and alignment of some conserved segments in every pair of matched blocks) is bounded by some function of d and r. Still it remains to determine the exact breaks. We call a pair of conserved segments of x and y an *aligned pair* if we have already decided to align them (as we did above in every pair of matched blocks). Now we are, in every branch, in a situation addressed by:

Lemma 1. *Consider an instance x, y of MCSP. Suppose that we have already fixed the following items:*
(i) exactly $k-1$ fragile segments in both x and y, which are pairwise disjoint, non-adjacent, do not include the first or last symbol of x or y, and have length at most n/k each,
(ii) a matching between the k conserved segments of x and y which are defined by cutting out these fragile segments,
(iii) an aligned pair in every matched pair of blocks given by (ii).
Then we can, in linear time, construct a solution to this constrained instance *of MCSP, or prove that no such solution exists.*

Proof. In the following, the terms *fragile segment* and *conserved segment* refer only to the $2(k-1)+2k$ distinguished segments specified in (i) and (ii). Since only $k-1$ breaks are allowed, every fragile segment contains exactly one break. The left (right) *semi-block* of a block is the segment between the member of the aligned pair and the left (right) break delimiting this block.

We define an auxiliary graph as follows. For every fragile segment f, we create two vertices that represent the two segments of f to the left and to the right of the break that will be placed in f. We call them the *left and right vertex* of f. We create edges that join
(1) the left and right vertex of every fragile segment,
(2) the left vertices of any two fragile segments in x and y whose conserved segments adjacent to the left are matched,
(2') the right vertices of any two fragile segments in x and y whose conserved segments adjacent to the right are matched.

The *weight* of a vertex is the length of the segment it represents. These weights are unknown in the beginning (as the breaks are yet unknown), but we get a linear equation for the weights of any two vertices joined by any edge, due to the following observations:
(1) The total length of any fragile segment is known.
(2) We have an aligned pair in the conserved segments to the left, and the number of symbols in their right semi-blocks must be the same in x and y.
(2'): similar to (2).

For a solution to the whole constrained problem instance, it is sufficient that:
(a) every symbol from x and y gets into exactly one block, and
(b) the matched semi-blocks to the left or right of aligned pairs have equal lengths, and are also equal as strings.

Equations from (1) ensure (a), and equations from (2) and (2') ensure the length condition in (b). Equality of strings in (b) must be tested separately, and not every solution to the linear system yields already a valid solution to the constrained MCSP instance. However, thanks to the structure of our linear system we need not search the entire solution space. Instead we do the following in every connected component of the graph.

We guess the weight of one vertex v. Then, the weights of all other vertices are uniquely determined by a trivial elimination procedure. Any solution specifies the lengths of certain semi-blocks. If no value of the weight of v yields a valid solution, i.e., if some weights become negative or some symbols that get aligned are not equal, then no valid solution can exist at all. On the other hand, since the semi-blocks to the left or right of an aligned pair are always associated to the same connected component, condition (b) is fulfilled if all aligned semi-blocks pass the equality test *in their respective connected components*. It follows that *any* combination of valid solutions from the connected components is a valid overall solution. Thus we can solve the subproblems and establish (a) and (b) in the connected components independently.

Analysis: We show the linear time bound. In an $O(n)$ time preprocessing in x and y we index the symbols from left to right, so that the length of any one segment can be computed by one subtraction in $O(1)$ time. Then the "graphical" linear system of equations can be set up in $O(k)$ time. Connected components are determined in $O(k)$ time as well. Once the weight of one vertex in a connected component of size k' is fixed, we can solve the linear system trivially in $O(k')$ time. The test whether the corresponding partitioning of x and y is valid can also be executed in $O(k')$ time, but this step needs some more preparation:

Since we have already fixed an aligned pair in every matched pair of blocks, we can easily determine the maximum number of symbols in any pair of aligned semi-blocks: Note that aligned symbols have to be equal. This gives a threshold for the number of symbols in every fragile segment that can be attached to the block to the left and right, respectively. For any solution to the linear system it suffices now to check whether each variable is nonnegative and below its threshold. All thresholds can be computed once in the beginning, in another $O(n)$ time preprocessing for the instance, just by symbol comparisons and counting.

In every connected component we need to try at most n/k possible weights of the start vertex, since this is the maximum length of a fragile segment. As we saw above, for every initial vertex weight we can solve the system and validate the solution in $O(k')$ time. Thus we need $O(k'n/k)$ time in the component. For all components together this sums up to $O(kn/k) = O(n)$. □

Theorem 1. *MCSP is fixed-parameter tractable in the combined parameters d, r (distance ratio and repetition number). More specifically, MCSP can be solved in $O(n)$ time for any fixed values of d, r.*

Proof. In the branching phase we fix the items specified in Lemma 1. The number of branches is bounded by a function of d, r. The assumptions of Lemma 1 are in fact satisfied: The selected fragile segments are pairwise disjoint by construction,

and $c \geq 2d$ ensures that none of them are adjacent or include the beginning or the end of x or y. Fragile segments have length $n/c \leq n/2d < n/k$. Now we solve every constrained instance in $O(n)$ time. Any solution found in a branch is a solution to the unconstrained instance, and if all branches are unsolvable, then so is the unconstrained instance. □

The scheme can be modified to efficiently compute a concise representation of *all* solutions to an instance of MSCP that comply with the given parameter values. We consider all branches anyhow (i.e., no potential solution is lost), and the solution space of the linear system of equations and inequalities (for nonnegativity, thresholds) can be concisely described.

The graphs in Lemma 1 have actually a very special structure, but we did not make use of this fact, because linear time is already optimal for trivial reasons. However, in practice one can save physical running time also by accelerating some parts of a linear-time algorithm, therefore we go into some more details now. Note that the graphs corresponding to our linear systems merely consist of even-length cycles. We call an edge, corresponding to an equation $u + v = a$ and $u - v = a$ (where u, v are variables and a some constant), an *additive and subtractive edge*, respectively. Now observe that, in every cycle, additive and subtractive edges alternate. Thus we can easily divide the vertices in two sides so that edges between vertices on different sides (on the same side) are additive (subtractive). If a linear system with two variables per equation, where each variable is integer and ranges from 0 to an individual threshold, has such a "quasi-bipartite" graph, we can find a solution in every connected component already by a logarithmic number of guesses of one variable: All variables on one side can only increase simultaneously, while all variables on the other side decrease. Hence we can find two extremal values of our probe variable (such that all other variables are still in their ranges) by binary search. As we said, this does not help the worst-case time bound, but it speeds up the search if the graph has only a few connected components.

3 Branching Heuristics

A closer look at the above FPT algorithm gives an explicit bound on the hidden factor in $O(n)$: The $k-1$ (out of $2d$) pieces in x with breaks can be chosen in $(2d-k)^{k-1}/(k-1)!$ ways, since no adjacent pieces have breaks. Blocks can be permuted in $k!$ ways. Since the block lengths are now fixed, up to a tolerance of 2 piece lengths, we can choose the pieces in y with breaks, successively from left to right, in 3^{k-1} ways. In every pair of matched blocks, at most r possible alignment partners exist. In total we get at most $kr(3(2d-k)r)^{k-1}$ branches. However, this is a crude bound. For typical, rather irregular input strings we expect that the vast majority of branches can be quickly discarded: Note that already one symbol mismatch in the segments we try to align renders a branch impossible. Therefore it is worthwhile to add to the basic algorithm some simple and fast heuristics that recognize many hopeless branches *before* we run the procedure of Lemma 1.

Some ideas have been already brought up in approximation algorithms [11], but not in the context of parameterized exact algorithms. To apply the linear-time procedure we need enough "well-separated" fragile segments (see Lemma 1 for details). A simple observation is that a segment of x not occuring as a substring in y must be fragile in x. We can search for all occurrences in y of a particular substring in $O(n)$ time, using suffix trees or linear-time string matching.

We suggest a top-down approach to find shorter and shorter fragile segments: Halve the string x recursively, and in the jth iteration check which minimal concatenations of the obtained 2^j pieces have no occurences in y. Moreover, from the obtained list F of fragile segments we can erase those containing shorter fragile segments detected later. Clearly, the time for this procedure remains bounded by n times a polynomial of distance ratio d. We need not even fix a parameter value d in advance, but we may also run the process until it succeeds with enough fragile segments, or some time limit is reached. There is a good chance that many long segments have only a few occurences in y: Intuitively, many occurences of many different segments must heavily overlap, hence they can exist only if y contains periodic segments with many repetitions. Therefore we should, in this way, quickly obtain $k-1$ disjoint fragile segments. (Here it is also good to notice that the maximum number of pairwise disjoint segments in a given set F of segments is easily determined by a linear-time greedy algorithm.) Next, for aligning pairs of conserved segments it is barely necessary to try all $k!$ matchings. For the same reason as above, our conserved segments should typically have few occurences in y, so that relatively few matchings remain.

We do not only keep the number of matchings small, but also the combinations of breaks, based on another simple fact: Once we know some set F of fragile segments, the set of breaks must be a hitting set of size at most $k-1$ for F. In particular, breaks can only lie in the union U of all hitting sets of size $k-1$. Often U will be considerably smaller than the segments covered by F, and consist of several disjoint segments (ideally $k-1$). On the other hand, we can compute U in linear time, so that the $O(n)$ time bound is not sacrificed. We give the details now. In the following, a "point" is a position between two symbols.

Theorem 2. *Given a set F of segments in a string of length n (none of them contained in another segment of F) and an integer k, we can compute the union U of all hitting sets of size $k-1$ for F in $O(n)$ time.*

Proof. First note that $|F| \leq n$. All segments in F not containing a specific point p are divided in two sets $LF(p)$ and $RF(p)$: the segments to the left and to the right of p. A minimum hitting set of $LF(p)$ is obtained by an obvious greedy algorithm that scans the points from left to right and always adds the latest possible point to the hitting set. This yields the size $lf(p)$ of a minimum hitting set for $LF(p)$, for *all* p together, in $O(n)$ time. Similarly we proceed with all $LR(p)$ and get all $lr(p)$. Finally observe that $p \in U$ if and only if $lf(p) + rf(p) \leq k-2$. □

-	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
x	c	a	b	c	c	d	a	a	a	a	b	a	b	c	c	a	b	a	b	d	c	d
y	b	d	c	d	c	a	b	c	c	a	b	a	b	d	a	a	a	c	c	a	b	a

We illustrate the heuristics on this small example taken from [5]. Refining the "grid" of segment endpoints on string x by successive halving, we find the following segments that do not appear in y and must be fragile: $[1, 22]$, $[1, 11]$, $[12, 22]$, $[1, 6]$, $[7, 11]$, $[17, 22]$, $[4, 6]$, $[9, 11]$, $[5, 7]$, $[7, 10]$, $[11, 14]$, $[18, 21]$, etc. Taking only the minimal fragile segments found until now, in left-to-right order, we retain the following set F: $[4, 6]$, $[5, 7]$, $[7, 10]$, $[9, 11]$, $[11, 14]$, $[18, 21]$. This F has 4 disjoint intervals, hence $k \geq 5$. We try whether $k = 5$ is enough. The simple algorithm from Theorem 2 restricts the possible breaks to segments $[5, 6]$, $[9, 10]$, $[11, 14]$, $[18, 21]$. That is, under the assumption $k \geq 5$ we obtained 4 disjoint fragile segments, where the 2 leftmost breaks are even uniquely determined. Next we align the conserved segments between the breaks with segments of y. Segments $[1, 5] = cabcc$, $[6, 9] = daaa$, and $[21, 22] = cd$ appear only once in y. Segment $[14, 18] = ccaba$ appears twice, but one such segment in y overlaps $cabcc$, hence this alignment is also unique. Now $[10, 11] = ab$ must be placed between $cabcc$ and $daaa$ in y. Here we have a case with repetition number 2, but we never had to branch on the permutations of blocks in this small example. Finally, the algorithm from Theorem 1 applied to these constellations finds the optimal solution $x = cabcc|daaa|abab|ccaba|bdcd$ and $y = bdcd|cabcc|abab|daaa|ccaba$.

4 Separating Close Breaks

Consider a constrained instance of MCSP with aligned pairs and fragile segments as in Lemma 1. The linear algorithm in Lemma 1 works only under the assumption that each of the designated fragile segments has exactly one break. Now we relax this assumption and suppose that, in each of the fragile segments, all breaks are within a segment of length w called a *window*. Here, w is the maximum window size in all fragile segments. The case solved by Lemma 1 is $w = 0$. In the following, *long blocks* of a partitioning of x, y are those containing the conserved segments that we identified so far, in the constrained instance. The others are *short blocks*. In order to apply a procedure similar to Lemma 1, we will need windows that have a uniform length t and are also aligned, therefore we have to enlargen the windows first.

Lemma 2. *If there is a solution to the MCSP instance constrained as above, then there exist windows in the fragile segments with the following properties:*
(1) All breaks are contained in the windows.
(2) All windows have exactly the same length $t < 2kw$.
(3) The long blocks in the partitioning are aligned in such a way that also the ends of windows are aligned. (But still the long blocks may end inside the windows.) Equivalently, symbols in windows are only aligned to symbols in windows in the other string.

Proof. We will only extend the given windows, to obtain properties (2) and (3). Thus, (1) is preserved automatically. First we extend every window to the left and to the right (if necessary) to establish (3). Since all long blocks end inside the given windows, we never have to move the end of a window by more than w symbols outwards. Thus we have amplified the windows to length at most $3w$. In the next phase we extend them further to equalize their lengths, without destroying (3). The difficulty is that any two windows from x and y whose left (right) ends are aligned must be extended simultaneously to the left (right).

Since (3) is already true, the sum of lengths of all windows in x and y are equal. We define a graph, with windows as vertices, where edges connect windows whose left or right ends are aligned. Every vertex is labeled with the length of the window it represents. Clearly, this graph is just a disjoint union of even cycles, and vertices come alternatingly from x and y. (The graph is related to the one in Lemma 1, but not identical.) Consider any cycle. We are allowed to add 1 to the labels of any pair of adjacent vertices. As long as two adjacent vertices have non-maximum labels, we simply increment them. If no vertices with non-maximal labels are adjacent, we choose some vertex with minimum label. Suppose this vertex represents a segment of x (the other case is symmetric). Because of the equal sums of labels in x and y, there must exist also a vertex from y with non-maximum label. The two chosen vertices have an odd distance in the cycle and split it in two paths, in the natural sense. Now it is easy to increase the labels of our two special vertices by 2, and the all other labels by only 1. Iterating the procedure we arrive at property (2). To bound the number of steps, consider the sum of differences between the maximum label and all vertex labels. The above procedure decreases this quantity by 2, and initially it was no larger than $2(k-1)w$. Hence the final window length is less than $2kw$. □

Theorem 3. *MCSP is fixed-parameter tractable in the combined parameters k, r (number of blocks and repetition number). More specifically, MCSP can be solved in $O(n \log n)$ time for any fixed values of k, r.*

Proof. We modify the linear system of Lemma 1 as follows: For every fragile segment, the sum of the two assigned variables *plus* t must equal the length of this fragile segment. Similarly as in Lemma 1 we solve the linear system and test whether some solution is *valid*, in the sense that the parts of long blocks *outside the windows* are aligned. At this stage we do not yet attempt to determine the breaks inside the windows, let alone the matching of short blocks.

First we set $t = 0$ and test whether a solution with exactly one break per fragile segment exists. If this fails, we need more breaks. In this case we determine the smallest $t > 0$ that gives a valid solution. A crucial observation for $t > 0$ is that the window positions in *all* valid solutions differ by some $O(t) = O(kw)$. Namely, if two valid solutions with remote window positions exist, then any two windows whose (e.g.) left ends are aligned must contain equal strings in one of the solutions, because these two windows are aligned segments within long blocks in the other solution. But then we can append these equal windows to the aligned long blocks to the left, and this can be done for all pairs of aligned

long blocks. But this means that one break per fragile segment would be enough, which contradicts the failure for $t = 0$. In conclusion, if a solution with window size w exists, there is *one* window of length $t := O(kw)$ in every fragile segment, so that all breaks are inside these windows, where t is the same everywhere. (We need not branch on $O(n)$ possible window positions!) The next difficulty is that we might have found a valid solution to the linear system for some $t > 0$, but a solution to the constrained MCSP instance exists only for some larger t, that is, some breaks are actually outside our windows. We have to analyze this situation.

Consider a partitioning where the total length of long blocks is maximal. In this partitioning, consider any pair of matched long blocks b, c which are not the rightmost blocks in x, y. Let u and v be the blocks next to the right of b and c, respectively. (The reasoning for the left-hand side is similar.) Assume $|u| \leq |v|$, the other case is symmetric. Consider the windows to the right of b and c. Note that the segments to the left of the window in x and y, until the long blocks, are equal substrings, and the left ends of u and v are aligned. Hence, if u is entirely to the left of the window, then u is a prefix of v. Now we can transform the partitioning as follows. We append u and the prefix of length $|u|$ of v to the long blocks. If $u = v$, and u, v are matched, then the number of blocks decreases. Otherwise, we split the matching partner of v in two blocks: the prefix of length $|u|$ and the (possibly empty) rest. The former is matched with the old alignment partner of u, and the latter (if not empty) with the rest of v. This yields a new partitioning, where the number of blocks is not increased, but the long blocks have larger total length, contradicting the maximality. Hence there exists a partitioning where, in every fragile segment, at most one break is to the left of the window we have determined. By symmetry, the same statement holds "to the right".

Now consider a fragile segment where the outermost breaks have the maximum distance (in all fragile segments), denoted w. Recall that our $t = O(kw)$ may be too small, but as shown above, we can assume that at most one break is to the left and to the right, respectively, of our window of length t in this fragile segment. If an outermost break is away from the window by more than, say, t symbols, there must still exist a segment of length $\Omega(t/k)$ without breaks, in the segment of length t next to the window, on the same side. The other case is that all breaks are still in an extended window of length $O(t)$. Since $w = \Omega(t/k)$, there must be a segment of length $\Omega(t/k^2)$ without breaks in the window.

Finally we are able to guess a window and therein a new conserved segment that separates two breaks. Due to the minimum distance guarantee for two of the breaks, the number of branches is bounded by a function of parameter k. We perform this branching step in both x and y, to guess at least one new aligned pair of conserved segments. Details are similar as in Section 2 (guessing pieces with breaks, all possible matchings, etc.). This step increases the number of blocks, hence, after less than k such branching steps we have finally separated all breaks. For any fixed value of the parameters, the time is $O(n \log n)$: We always find the smallest suitable t by binary search, solving $O(\log n)$ times a system of equations and checking validity in $O(n)$ time. □

5 MCIP with a Fixed Number of Integers

MCIP appears as a subcase of MCSP, in a sense: It is not hard to transform any instance of MCIP to an equivalent instance of MCSP where the integers are represented as periodic strings of those lengths. (However, this is not a polynomial transformation, since integers are represented unary.) The obtained MCSP instances have large repetition numbers, a case that is not captured by our FPT results. Therefore it is interesting to notice the following "pseudo FPT" result:

Theorem 4. *For any fixed cardinality of multisets, MCIP can be solved in polynomial time.*

Proof. As observed in [3], MCIP is equivalent to the following problem: Partition both multisets into the same number of sub-multisets, as *many* as possible, and match them one-to-one, so that the integers in any matched pair have the equal sums. From any such partition of the multisets, one can construct in polynomial time an optimal solution to the MCIP instance, by a simple elimination process. (Due to lack of space we omit the straightforward details of this equivalence.)

It remains to find a pair of partitionings that can be matched. The number of partitionings is obviously bounded by a function of cardinality of multisets (by the Stirling number, to be specific). We may naively generate all partitionings of both multisets, and check whether a pair of them have the same sums of sub-multisets. The last step can be executed in polynomial time (in the number of partitionings), with help of lexicographic sorting. □

Due to the above correspondence to repetition numbers, this gives hope that MCSP might be fixed-parameter tractable in k as the only parameter: Theorem 4 might be applied, inside some sophisticated extension of the previous FPT algorithms, to any substring with many repeats in x, y. But we have to leave this as an open question.

6 Conclusions and Further Research

We proved that MCSP is fixed-parameter tractable in some natural parameters, indicating that MCSP is exactly solvable in practice. A simple algorithm needs distance ratio d (length ratio of blocks) and repetition number r of periodic segments as parameters, a second algorithm with weaker parameters k (number of blocks) and r is more intricate. We also proposed some simple heuristics that probably discard the vast majority of conceivable branches in most instances. Still it would be interesting to prove good worst-case bounds on the number of branches, using deeper combinatorics of strings. An intriguing open question is whether MCSP is fixed-parameter tractable in k. Since repeats are quite possible, it would be nice to get rid of parameter r. Implementing the FPT algorithms and testing their time performance of on real biological strings would complement the theoretical findings. A natural extension of MCSP is to tolerate a limited number of mismatches in the solutions. Apparently our FPT scheme can nicely accommodate this extra parameter: In the validity tests of solutions we just have to count the mismatches.

References

1. M.D.V. Braga, M.F. Sagot, C. Scornavacca, E. Tannier. The solution space of sorting by reversals, *3rd Int. Symposium on Bioinformatics Research and Applications ISBRA 2007*, *LNCS* 4463, 293-304, journal version to appear in *IEEE/ACM Transactions on Computational Biology and Bioinformatics*
2. L. Cai, X. Huang, C. Liu, F. Rosamond, Y. Song. Parameterized complexity and biopolymer sequence comparison, *The Computer Journal*, Special Issue in Parameterized Complexity, 51 (2008), 270-291
3. X. Chen, L. Liu, Z. Liu, T. Jiang. On the minimum common integer partition problem, *6th Italian Conference Algorithms and Complexity CIAC 2006*, *LNCS* 3998, 236-247
4. X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, T. Jiang. Assignment of orthologous genes via genome rearrangement, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2 (2005), 302-315
5. M. Chrobak, P. Kolman, J. Sgall. The greedy algorithm for the minimum common string partition problem, *7th Int. Workshop on Approximation Algorithms for Combinatorial Optimization Problems APPROX 2004*, *LNCS* 3122, 84-95
6. R.G. Downey, M.R. Fellows. *Parameterized Complexity*, Springer, 1999
7. A. Goldstein, P. Kolman, J. Zheng. Minimum common string partitioning problem: Hardness and approximations, *15th Int. Symposium on Algorithms and Computation ISAAC 2004*, *LNCS* 3341, 473-484, also in: *The Electronic Journal of Combinatorics* 12 (2005), paper R50
8. D. He. A novel greedy algorithm for the minimum common string partition problem, *3rd Int. Symposium on Bioinformatics Research and Applications ISBRA 2007*, *LNCS* 4463, 441-452
9. F. Hüffner, R. Niedermeier, S. Wernicke. Developing fixed-parameter algorithms to solve combinatorially explosive biological problems. In: *Bioinformatics, vol. II: Structure, Function and Applications*, vol. 453 in *Methods in Molecular Biology*, Humana Press, 2008.
10. P. Kolman. Approximating reversal distance for strings with bounded number of duplicates in linear time, *30th Int. Symposium on Mathematical Foundations of Computer Science MFCS 2005*, *LNCS* 3618, 580-590
11. P. Kolman, T. Walén. Reversal distance for strings with duplicates: Linear time approximation using hitting set, *The Electronic Journal of Combinatorics* 14 (2007), paper R50, also in: *4th Workshop on Approximation and Online Algorithms WAOA 2006*, *LNCS 4368*, 281-291
12. D.P. Woodruff. Better approximations for the minimum common integer partition problem, *9th Int. Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2006*, *LNCS* 4110, 248-259