JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 22, Number 6, 2015
© Mary Ann Liebert, Inc.
Pp. 577–594
DOI: 10.1089/cmb.2014.0166

Reconstructing Breakage Fusion Bridge Architectures Using Noisy Copy Numbers

SHAY ZAKOV and VINEET BAFNA

ABSTRACT

The Breakage Fusion Bridge (BFB) process is a key marker for genomic instability, producing highly rearranged genomes in relatively small numbers of cell cycles. While the process itself was observed during the late 1930s, little is known about the extent of BFB in tumor genome evolution. Moreover, BFB can dramatically increase copy numbers of chromosomal segments, which in turn hardens the tasks of both reference-assisted and *ab initio* genome assembly. Based on available data such as Next Generation Sequencing (NGS) and Array Comparative Genomic Hybridization (aCGH) data, we show here how BFB evidence may be identified, and how to enumerate all possible evolutions of the process with respect to observed data. Specifically, we describe practical algorithms that, given a chromosomal arm segmentation and noisy segment copy number estimates, produce all segment count vectors supported by the data that can be produced by BFB, and all corresponding BFB architectures. This extends the scope of analyses described in our previous work, which produced a single count vector and architecture per instance. We apply these analyses to a comprehensive human cancer dataset, demonstrate the effectiveness and efficiency of the computation, and suggest methods for further assertions of candidate BFB samples. Source code of our tool can be found online.

Key words: algorithms, combinatorial proteomics, computational molecular biology, dynamic programming, genetic variation, RNA, sequence analysis.

1. INTRODUCTION

The origin of a tumor cell is marked by genomic instability (Hanahan and Weinberg, 2011). Spontaneous, viral, or other kinds of mechanisms may cause genomic segment deletions, duplications, translocations, inversions, etc., producing rearranged genomes with a possibly malignant nature. Thus, decoding mechanisms that generate rearranged genomes is critical to understanding cancer. Numerous mechanisms were proposed, including the faulty repair of double-stranded DNA breaks by recombination or end-joining and polymerase hopping caused by replication fork collapse (Carr et al., 2011; Hastings et al., 2009). These mechanisms are generally not directly observable, so their elucidation requires the deciphering of often subtle clues after genomic instability has ceased. An important source of information in this respect is the *architecture* of the rearranged genome, that is, the description of its chromosomes in terms of concatenations of segments from the original genome.

Department of Computer Science and Engineering, University of California San Diego, La Jolla, California.

Breakage Fusion Bridge (BFB) is one model of a genome rearrangement process, which was first proposed by Barbara McClintock in the 1930s (McClintock, 1938, 1941). Recently, it has seen renewed interest as a possible mechanism in tumor genome evolution (Bignell et al., 2007; Campbell et al., 2010; Greenman et al., 2012). BFB begins with a telomeric loss on a chromosome, including a loss of a sequential pattern that signals the location of chromosome termination. During cell division the telomere-lacking chromosome replicates, and its two sister chromatids fuse together (possibly due to some DNA repair mechanism falsely induced by the cell). This fusion produces a dicentric chromosome of palindromic structure, which is later torn apart at some random point as the centromeres of the dicentric chromosome migrate to opposite poles of the cell. One part of the torn chromosome includes the fusion region and some tandemly inverted chromosomal suffix duplication, and the other part lacks the corresponding suffix. The two daughter cells receive these rearranged chromosomes, both are missing the telomeric region, and the cycle can repeat (Fig. 1).

In contrast to other mechanisms, BFB can actually be observed in progress using methods that have been available for decades (McClintock, 1941). Cytogenetic techniques can reveal the anaphase bridges, dicentric chromosomes, and homogeneously staining regions that have long been the canonical evidence for BFB. However, these techniques are useful only in cases where the BFB cycles are ongoing. While useful in understanding the mechanism, they do not address the question of whether BFB occurs extensively in evolving tumor genomes.

Recently, researchers (including us) have started looking at modern available data in order to demonstrate BFB occurrence after the process has ceased, including Fluorescent *In Situ* Hybridization (FISH), Array Comparative Genomic Hybridization (aCGH), and Next Generation Sequencing (NGS) data. These methods take advantage of distinctive BFB features exposed by such data, including the abundance of fold-back inversions (i.e., duplicated chromosomal segments arranged in a head-to-head orientation (Bignell et al., 2007; Campbell et al., 2010), patterns of interleaving segments of alternating orientations (Kitada and Yamasaki, 2008; Reshmi et al., 2007), and combinatorial properties of segment counts when copy number variations are due to BFB (Kinsella and Bafna, 2012; Zakov et al., 2013). In fact, if the architecture of the rearranged genome is known, it is possible to decide if this architecture can be produced by BFB (Kinsella and Bafna, 2012). Properties of the space of different BFB evolutions are explored in Greenman et al. (2012).

Partial knowledge regarding the architecture can be revealed by FISH analyses (Kitada and Yamasaki, 2008), which uses fluorescence markers to identify the physical locations of predetermined sequences on the rearranged genome. However, such experiments are relatively expensive and can only be performed in a small number of cases. A more common measurement is NGS data, which contain a big set of short sequenced reads extracted from a donor genome. Such data is typically used for predicting the entire donor genomic sequence by computationally assembling the reads, sometimes facilitated by consulting a similar presequenced reference genome. Unfortunately, BFB and other mechanisms can produce massively

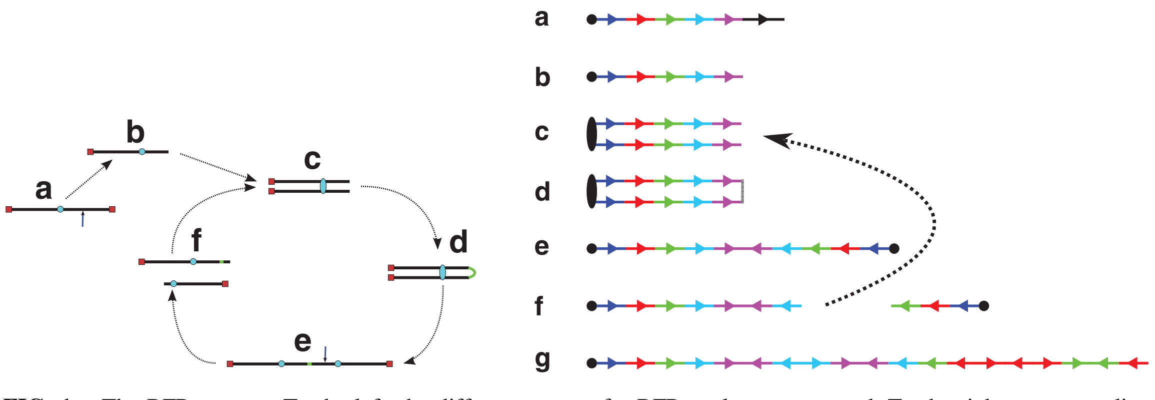

FIG. 1. The BFB process. To the left, the different stages of a BFB cycle are presented. To the right, corresponding modifications over an exemplary chromosomal arm are shown. (**a**) A normal chromosome. (**b**) The chromosome loses its telomere. (**c**) The chromosome is duplicated during cell division. (**d**) Sister chromatids are fused together. (**e**) Centromeres migrate to opposite poles of the cell. (**f**) The fused chromosome is torn apart at some random position between the two centromeres, causing one copy to have an inverted suffix duplication, while the other copy has a trimmed suffix. Both copies lack a telomere and therefore may undergo additional BFB cycles. (**g**) After several BFB cycles, the chromosome architecture exhibits significant increases in segment copy numbers, as well as fold-back patterns.

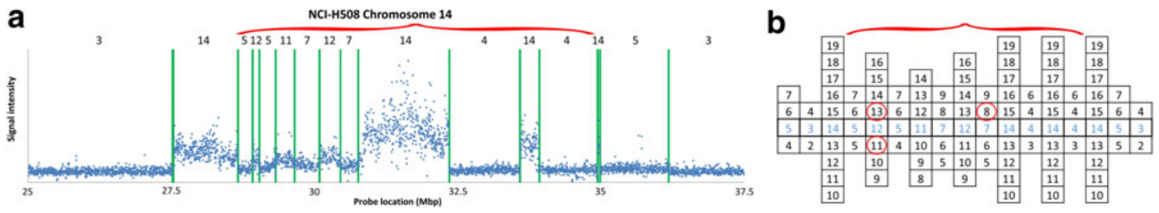

FIG. 2. (**a**) aCGH data for a part of the q-arm of human chromosome 14 in the NCI-H508 cell line. Each data point corresponds to a probe on the array, where its x-coordinate gives the probe's sequence chromosomal position, and y-coordinate gives its measured intensity (log-ratio). The data points are clustered into segments, and an estimated segment copy number appears above each segment. (**b**) A visualization of the corresponding noisy copy number data. Estimated counts appear in light blue, and the column around each count represents possible deviations from the estimation. The region under the red curly bracket reflects a BFB candidate. Its corresponding estimated counts are [5, 12, 5, 11, 7, 12, 7, 14, 4, 14, 4], which under minor modifications yield the two BFB vectors [5, **13**, 5, 11, 7, 12, **8**, 14, 4, 14, 4] or [5, **11**, 5, 11, 7, 12, **8**, 14, 4, 14, 4]. Data is taken from Bignell et al. (2010) [segmentation and copy number analysis were computed using the PICNIC software (Greenman et al., 2010)].

rearranged and highly repetitive genomes. This complicates the task of assembly-based sequencing due to the multiple ambiguous manners the repetitive reads may be assembled, and the lack of a relevant reference template. Nevertheless, NGS data can still be analyzed in order to infer some indirect information regarding the donor genome architecture (Alkan et al., 2009; Chiang et al., 2009; Medvedev et al., 2009; Yoon et al., 2009). After aligning the reads against a reference genome, their genomic location distribution can be used in order to identify segments on the reference genome of coherent read coverage, and to estimate the number of times each such segment repeats in the donor genome. We will refer to the output of the latter kind of analysis as *copy number data*. Other methods to obtain copy number data are based on analyzing aCGH data (Eckel-Passow et al., 2011; Greenman et al., 2010; Olshen et al., 2004; Venkatraman and Olshen, 2007) (Fig. 2). Due to the noisy nature of both NGS and aCGH data, count estimates may be inaccurate, and the true segment count is likely to fall within some interval of integers around the estimated value. We use the term *noisy copy number data* when referring to information regarding such intervals of possible count values. In addition to copy number data, NGS data can be used in order to produce contigs (chromosomal segments that may be assembled unambiguously), and aberrant segment adjacencies can be exposed by discordant reads, restricting the set of possible contig-based architectures.

In previous work (Kinsella and Bafna, 2012; Zakov et al., 2013), we showed how to analyze noisy copy number data in order to decide if it is likely to observe the input data under the assumption that the underlying rearrangement process is BFB. Specifically, we designed algorithms that produce a single BFB architecture over the given segments in which segment counts are supported by the data, if such an architecture exists. We applied these algorithms in order to analyze a comprehensive aCGH dataset of cancer cell lines (Bignell et al., 2010), as well as sequence data from primary tumors (Campbell et al., 2010), and identified a small subset of candidate samples exhibiting BFB hallmarks. Here, we extend the scope of the analysis and describe algorithms that report *all* count settings supported by the data, which can be explained by BFB, and all corresponding BFB architectures. Although the theoretical time bounds for these new algorithms may be exponential, we show that in practice they are efficient and apply an informed search (Pearl, 1984) optimization that further improves their practical efficiency.

Therefore, our proposed algorithms satisfy an important need. While our work postulates the existence of BFB using statistical arguments, additional physical assertions can be obtained with FISH and aberrant read analyses. Starting with noisy copy number data, our tool can be used to enumerate all possible BFB architectures. These candidate architectures can then be used toward a small set of FISH experiments (with a limited number of fluorescence markers) to validate and refine the predicted genomic architecture.

2. PROBLEM DEFINITION

Computational BFB-related problems were previously formulated in Kinsella and Bafna (2012) and Zakov et al. (2013). For completeness, we give here the main definitions from these works and formulate new problems first addressed here.

A DNA segment σ is a string over the DNA nucleotide alphabet A, C, G, T. The reversed segment of a segment σ, denoted here by $\overline{\sigma}$, is the string obtained by reading σ backwards and replacing each nucleotide with its complementary nucleotide ($A \leftrightarrow T$, $C \leftrightarrow G$). For example, the reverse of a segment $\sigma = CGGAT$ is the segment $\overline{\sigma} = ATCCG$. In the rest of this article, it is assumed we operate on a given chromosomal arm with a fixed segmentation and denote its list of k segments by $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$, ordered from the centromeric segment σ_1 to the telomeric segment σ_k. The term "string" refers to a genomic architecture over these segments, that is, a concatenation of segments from Σ and their reversed forms. Greek letters α, β, γ, and ρ denote strings, and bar notation indicates reversed strings. For example, if $\alpha = \sigma_1\sigma_3\overline{\sigma}_2$, $\overline{\alpha} = \sigma_2\overline{\sigma}_3\overline{\sigma}_1$. An empty string is denoted by ε. The notation $\alpha_{l,t}$ represents the continuous chromosomal region $\sigma_l\sigma_{l+1}\ldots\sigma_t$, where $\alpha_{l,t} = \varepsilon$ when $t < l$. To facilitate reading, $\sigma_1, \sigma_2, \sigma_3, \ldots$ are replaced by A, B, C, $\ldots$ in concrete examples.

A BFB cycle applied over a chromosomal arm can be viewed as a special rearrangement procedure, in which some telomeric suffix of the arm is duplicated, inverted, and concatenated tandemly at the telomeric end of the arm. A string β can be derived from a string α via a BFB process if it is possible to apply a series of zero or more BFB cycles over α and obtain β (Fig. 3a). This notion is formally captured by the following definition.

Definition 1 *For two strings α, β, say that $\alpha \xrightarrow{BFB} \beta$ if $\alpha = \beta$, or there are some strings ρ, γ such that $\gamma \neq \varepsilon$, $\alpha = \rho\gamma$, and $\rho\gamma\overline{\gamma} \xrightarrow{BFB} \beta$. Say that α is an l-BFB string if $\alpha_{l,t} \xrightarrow{BFB} \alpha$ for some t, and say that α is a BFB string if it is an l-BFB string for some l.*

It is worth mentioning that in reality a BFB cycle can also delete a suffix of a chromosome in case we consider the trimmed chromosomal arm. It is simple to show that for any BFB process that contains such suffix-trimming cycles there is an equivalent process in which only elongation cycles occur (Kinsella and Bafna, 2012). For example, the BFB string $\mathrm{ABCD\overline{D}}$ obtained by the process $\alpha_{1,4} = \mathrm{ABCD} \xrightarrow{\mathrm{BFB}} \mathrm{ABCD}\mathbf{\overline{D}\overline{C}\overline{B}} \xrightarrow{\mathrm{BFB}} \mathrm{ABCD\overline{D}}$ can also be obtained by the process $\alpha_{1,4} = \mathrm{ABCD} \xrightarrow{\mathrm{BFB}} \mathrm{ABCD}\mathbf{\overline{D}}$. Thus, we can assume without loss of generality that all BFB cycles are of the form of a tandem suffix duplication.

By definition $\varepsilon = \alpha_{l,l-1}$ is an l-BFB string for every $l \geq 1$. For a nonempty string α, define top(α) = max $\{t : \sigma_t$ appears in $\alpha\}$ and define top(ε) = 0. It is simple to observe that when α is an l-BFB string, it must start with the prefix $\alpha_{l,t}$ for t = top(α), since BFB cycles can only duplicate previously appearing letters and never generate new ones.

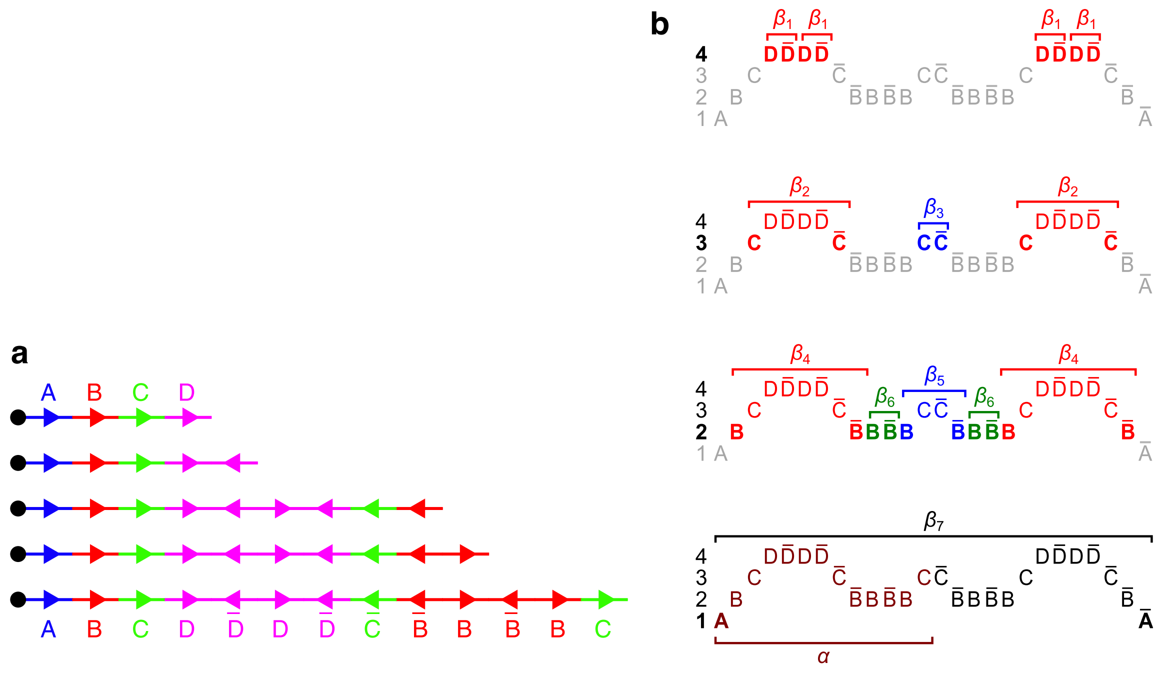

FIG. 3. (**a**) A BFB process generating a string α: $\mathrm{ABCD} \xrightarrow{\mathrm{BFB}} \mathrm{ABCD}\mathbf{\overline{D}} \xrightarrow{\mathrm{BFB}} \mathrm{ABCD\overline{D}}\mathbf{D\overline{D}\overline{C}\overline{B}} \xrightarrow{\mathrm{BFB}} \mathrm{ABCD\overline{D}D\overline{D}\overline{C}\overline{B}}\mathbf{B} \xrightarrow{\mathrm{BFB}} \mathrm{ABCD\overline{D}D\overline{D}\overline{C}\overline{B}B}\mathbf{\overline{B}}\mathrm{BC}$. (**b**) The layers of the BFB palindrome $\beta = \alpha\overline{\alpha}$. The block collections are $B^4 = \{4\beta_1\}$, $B^3 = \{2\beta_2, \beta_3\}$, $B^2 = \{2\beta_4, \beta_5, 2\beta_6\}$, and $B^1 = \{\beta_7\}$.

The *count vector* $\vec{n}(\alpha) = [n_1, n_2, \ldots, n_k]$ of a string α is a vector of integers, where for every $1 \leq l \leq k$, n_l is the total number of occurrences of σ_l and $\overline{\sigma}_l$ in α. For example, for $\alpha = \text{ABCD}\bar{\text{D}}\bar{\text{C}}\text{C}$, $\vec{n}(\alpha) = [1, 1, 3, 2]$. Say that a vector $\vec{n}$ is a *BFB vector* if there exists some BFB string α such that $\vec{n} = \vec{n}(\alpha)$. In the previous example $\vec{n}(\alpha)$ is a BFB vector due to the BFB process $\alpha_{1,4} = \text{ABCD} \overset{\text{BFB}}{\longrightarrow} \text{ABCD}\bar{\mathbf{D}}\bar{\mathbf{C}} \overset{\text{BFB}}{\longrightarrow} \text{ABCD}\bar{\text{D}}\bar{\text{C}}\mathbf{C} = \alpha$.

The computational analyses presented in this article aim to detect evidence for BFB, given a preanalyzed segmentation of the genome and corresponding copy number data. We assume that noisy copy number data is represented by a *weight function* $W = \{w_{l,n} \mid 1 \leq l \leq k, n = 0, 1, 2, \ldots\}$, where $w_{l,n}$ is a nonnegative *weight* associated with the copy number n for the l-th segment. It may be assumed w.l.o.g. that all weights $w_{l,n}$ satisfy $0 \leq w_{l,n} \leq 1$. The *weight* of a count vector $\vec{n} = [n_1, n_2, \ldots, n_k]$ is given by $W(\vec{n}) = \prod_{1 \leq i \leq k} w_{i, n_i}$, and by assumption $0 \leq W(\vec{n}) \leq 1$. In some cases, we refer to prefixes $\vec{n}_{1, l-1} = [n_1, n_2, \ldots, n_{l-1}]$ and suffixes $\vec{n}_{l,k} = [n_l, n_{l+1}, \ldots, n_k]$ of $\vec{n}$, which may be empty if $l = 1$ or $l = k + 1$, respectively. Define the weights of such subvectors accordingly, that is, $W(\vec{n}_{1,l-1}) = \prod_{1 \leq i < l} w_{i, n_i}$ and $W(\vec{n}_{l,k}) = \prod_{l \leq i \leq k} w_{i, n_i}$, where the weight of an empty vector is 1 by definition. Thus, for every $1 \leq l \leq k+1$, $W(\vec{n}) = W(\vec{n}_{1,l-1}) \cdot W(\vec{n}_{l,k})$.

If some data analysis produces segment count probabilities $\Pr(n_l = n)$ for every segment σ_l and every count $n = 0, 1, 2, \ldots$, weights can be set to these probabilities choosing $w_{l,n} = \Pr(n_l = n)$. This way, the weight of a count vector is the probability this vector reflects the true segment counts given the observed data. Another way to set weights given such probabilities would be to choose weights by setting $w_{l,n} = \frac{\Pr(n_l = n)}{\Pr(n_l = n_l^*)}$, where n_l^* is the most likely count for the l-th segment. Here, the weight of a count vector gives the ratio between its probability and the probability of a most likely vector. Nevertheless weights are more general than probabilities and can be used as a heuristic count error modeling even when no probabilistic model is available.

In Zakov et al. (2013), several variants of BFB problems were formulated. Below we restate these problems and add two new variants addressed in the current work:

BFB problem variants

Input: *A count vector $\vec{n}$, or a weight function W and a minimum weight threshold* $0 < \eta \leq 1$.

1. **The decision variant** (Zakov et al., 2013): *given $\vec{n}$, decide if $\vec{n}$ is a BFB vector.*
2. **The string search variant** (Zakov et al., 2013): *if $\vec{n}$ is a BFB vector, find a BFB string α such that $\vec{n} = \vec{n}(\alpha)$.*
3. **The vector search variant** (denoted **the distance variant** in Zakov et al., 2013): *given W and η, report a maximum weight BFB vector $\vec{n}$ in case there exists such a vector with $W(\vec{n}) \geq \eta$, and otherwise report "FAILED."*
4. **The exhaustive vector search variant:** *given W and η, report all BFB vectors $\vec{n}$ with $W(\vec{n}) \geq \eta$.*
5. **The exhaustive string search variant:** *given W and η, report all BFB strings α such that $W(\vec{n}(\alpha)) \geq \eta$.*

For a count vector $\vec{n}$, define $N(\vec{n}) = \sum_{1 \leq l \leq k} n_l$ and $\tilde{N}(\vec{n}) = \sum_{1 \leq l \leq k} \log(n_l)$. Note that $N(\vec{n})$ is the total length of a string admitting $\vec{n}$, and $\tilde{N}(\vec{n})$ is proportional to the number of bits needed for representing $\vec{n}$. For a weight function W and a weight η, define $N(W, \eta) = \max\{N(\vec{n}) : W(\vec{n}) \geq \eta\}$, and $\tilde{N}(W, \eta) = \max\{\tilde{N}(\vec{n}) : W(\vec{n}) \geq \eta)\}$. In Zakov et al. (2013), it was shown that the BFB decision variant can be solved using $O(\tilde{N}(\vec{n}))$ bit operations (i.e., linear time in the input length), the string search variant can be solved in $O(N(\vec{n}))$ operations (i.e., linear time in the output length), and that the vector search variant can be solved using at most a subexponential number of operations $2^{O(\log 2 \; N(W,\eta))}$. Here, we give algorithms for the two new exhaustive search variants. While theoretically the output of these algorithms can be exponential with respect to $N(W, \eta)$, we show that for realistic inputs this output is manageable. In addition, we describe an Informed Search (IS) approach that significantly reduces the running time in practice by eliminating irrelevant search paths and traversing only paths that are guaranteed to produce valid solutions.

3. ALGORITHMS

In this section we develop algorithms for the two exhaustive search variants of the BFB problem. Next, we describe some ideas taken from Zakov et al. (2013), upon which the algorithms presented here are built.

3.1. Notation and previous results

An *l-BFB palindrome* is an l-BFB string of the form $\beta = \alpha\bar{\alpha}$. It can be shown that $\beta = \alpha\bar{\alpha}$ is an l-BFB palindrome if and only if α is an l-BFB string. By definition, $\varepsilon = \varepsilon\bar{\varepsilon}$ is an l-BFB palindrome for every $l \geq 1$.

In addition, observe that when $\beta = \alpha\bar{\alpha}$ we have that $\vec{n}(\beta) = 2\vec{n}(\alpha)$. This allows replacing the question ''is there a BFB string admitting the count vector $\vec{n}$'' by the equivalent question ''is there a BFB palindrome admitting the count vector $2\vec{n}$''.

An *l-block* is a string of the form $\beta = \sigma_l\beta'\overline{\sigma}_l$, where β' is an $(l+1)$-BFB palindrome. It can be shown that an l-block is a special form of an l-BFB palindrome, and that every l-BFB palindrome is some palindromic concatenation of l-blocks. Nevertheless, not every palindromic concatenation of l-blocks yields a valid l-BFB palindrome. For example, two copies of the 2-block $\mathrm{BC\bar{C}\bar{B}}$ and one copy of the block $\mathrm{B\bar{B}}$ can be concatenated to form the 2-BFB palindrome $\mathrm{BC\bar{C}\bar{B}}\mathbf{B\bar{B}}\mathrm{BC\bar{C}\bar{B}}$. The validity of this palindrome can be asserted from the process $\alpha_{2,3} = \mathrm{BC} \overset{\mathrm{BFB}}{\longrightarrow} \mathrm{BC}\mathbf{\bar{C}\bar{B}} \overset{\mathrm{BFB}}{\longrightarrow} \mathrm{BC\bar{C}\bar{B}}\mathbf{B} \overset{\mathrm{BFB}}{\longrightarrow} \mathrm{BC\bar{C}\bar{B}B}\mathbf{\bar{B}BC\bar{C}\bar{B}}$. On the other hand, the only palindromic concatenation of one copy of $\mathrm{BC\bar{C}\bar{B}}$ and two copies of $\mathrm{B\bar{B}}$ is the string $\mathrm{B\bar{B}}\mathbf{BC\bar{C}\bar{B}}\mathrm{B\bar{B}}$. This string is not a valid BFB string, since a BFB string over the letters {B, C} must start with the prefix $\alpha_{2,3} = \mathrm{BC}$. Claim 1 in Zakov et al. (2013), recited here in the Appendix, gives a required and sufficient condition for block concatenations that form valid BFB palindromes.

The idea of decomposing BFB palindromes into blocks allows us to adopt a layeresd view of BFB palindromes, as follows (Fig. 3). Let $\beta = \alpha\bar{\alpha}$ be a 1-BFB palindrome, where $\vec{n}(\beta) = 2\vec{n}(\alpha) = [2n_1, 2n_2, \ldots, 2n_k]$. As claimed above, β is a palindromic concatenation of 1-blocks. Denote by B^1 the collection of all 1-blocks whose concatenation forms β. Every 1-block in B^1 is a string of the form $\mathrm{A}\beta'\bar{\mathrm{A}}$, where β' is some 2-BFB palindrome. As there are $2n_1$ occurrences of A and $\bar{\mathrm{A}}$ in β, and each block in B^1 contains exactly two such occurrences, the total number of blocks in B^1 is exactly n_1. Masking the letters A and $\bar{\mathrm{A}}$ from all blocks in B^1, the collection becomes a 2-BFB palindrome collection of size n_1. The 2-BFB palindromes in this collection can be further decomposed into two-blocks, yielding a collection B^2 of two-blocks. Similarly as above, B^2 contains exactly n_2 blocks. This process can continue inductively, yielding for every $1 \le l \le k$ a corresponding collection B^l of l-blocks, whose size is n_l. One may also imagine an additional collection in this series B^{k+1}, containing zero $(k+1)$-blocks.

This layered view is exploited in a reversed order by the algorithms in Zakov et al. (2013), developing a BFB palindrome given an input count vector $\vec{n} = [n_1, n_2, \ldots, n_k]$: Starting with an empty collection B^{k+1} of $(k+1)$-blocks, the algorithm computes iteratively a sequence of collections $B^k, B^{k-1}, \ldots, B^1$, each collection B^l is an l-block collection of size n_l. In order to generate B^l, the algorithm first concatenates $(l+1)$-blocks from B^{l+1}, forming a collection B of $(l+1)$-BFB palindromes of size n_l (this procedure is called folding). Then, each $(l+1)$-BFB palindrome $\beta' \in B$ is wrapped with a pair of σ_l segments, rendering it into an l-block $\beta = \sigma_l\beta'\overline{\sigma}_l$, and B^l is set to be the collection containing all these l-blocks. The final collection of 1-blocks B^1 is folded one more time into a single 1-BFB palindrome $\beta = \alpha\bar{\alpha}$, and the algorithm returns the half-length prefix α of this palindrome as a BFB string admitting the input count vector $\vec{n}$.

Figure 3b illustrates a possible run of the algorithm over the input count vector $\vec{n} = [1, 5, 3, 4]$. First, the algorithm initializes an empty collection of blocks B^5. In the first iteration, there is a need to perform concatenations of blocks in B^5 and produce $n_4 = 4$ BFB palindromes. Such palindromes may only be obtained by concatenating zero elements (as there are no elements in B^5), and so four empty strings are generated in this folding process, yielding the BFB palindrome collection $\{4\varepsilon\}$. Next, each palindrome in this collection is wrapped by $\sigma_4 = \mathrm{D}$ and $\overline{\sigma}_4 = \bar{\mathrm{D}}$, producing the collection of blocks $B^4 = \{4\mathrm{D}\varepsilon\bar{\mathrm{D}}\} = \{4\beta_1\}$. In the next iteration, the collection B^4 needs to reduce its size from $n_4 = 4$ into $n_3 = 3$ by concatenating its elements to produce BFB palindromes. In this example, there are two concatenations of two elements the form $\beta_1\beta_1$, and one concatenation of zero elements that produces an empty string ε. The BFB palindromes in the resulting folded collection $\{2\beta_1\beta_1, \varepsilon\}$, are wrapped by $\sigma_3 = \mathrm{C}$ and $\overline{\sigma}_3 = \bar{\mathrm{C}}$, yielding the block collection $B^3 = \{2\mathrm{C}\beta_1\beta_1\bar{\mathrm{C}}, \mathrm{C}\varepsilon\bar{\mathrm{C}}\} = \{2\beta_2, \beta_3\}$. This process continues for two more iterations, generating similarly the collections $B^2 = \{2\beta_4, \beta_5, 2\beta_6\}$ and $B^1 = \{\beta 7\}$. All elements in the last collection B^1 are then concatenated into a single BFB palindrome β (in this example B^1 contains a single element β_7, and so $\beta = \beta_7$), and the returned string α is the half-length prefix of this palindrome.

The ability of the schematic algorithm above to process the entire input vector $\vec{n}$ and produce a corresponding BFB string depends on its ability to fold intermediate collections B^l computed along its run. In cases where it cannot fold some intermediate block collection, it returns a fail message, implying no BFB string admits the input vector $\vec{n}$.

A case where folding cannot be applied is, for example, the case where $n_2 = 2$, $B^2 = \{\mathrm{BC\bar{C}\bar{B}}, \mathrm{B\bar{B}}\}$, and $n_1 = 1$. In this case, since both possible concatenations $\mathrm{BC\bar{C}\bar{B}}\mathbf{B\bar{B}}$ and $\mathbf{B\bar{B}}\mathrm{BC\bar{C}\bar{B}}$ of the two elements in B^2 are non-palindromic, the folding procedure must fail at this stage. Another example of a fail folding is the case where $n_2 = 3$, $B^2 = \{\mathrm{BC\bar{C}\bar{B}}, 2\mathrm{B\bar{B}}\}$, and $n_1 = 1$. In this case, though there exists a palindromic concatenation $\mathbf{B\bar{B}}\mathrm{BC\bar{C}\bar{B}}\mathbf{B\bar{B}}$ of all three elements in B^2, this concatenation is not a valid BFB palindrome (see example above), and so the collection may not be folded.

In Zakov et al. (2013), it was shown that the ability to fold a block collection depends on a property called the *signature* of the collection. A signature $\vec{s} = \vec{s}(B)$ of an l-BFB palindrome collection B is an infinite sequence of integers $[s_0, s_1, s_2, \ldots]$ with the following properties: (1) the first nonzero element in $\vec{s}$ (if there is such an element) must be positive, (2) the *cardinality* of $\vec{s}$, defined by $\|\vec{s}\| = \sum_{d=0}^{\infty} 2^d \mathrm{abs}(s_d)$ (where $\mathrm{abs}(s_d)$ is the absolute value of s_d), equals the size of B, and (3) the values s_d depend only in multiplicities of distinct elements in B and their top values. The prefix of a signature $\vec{s}$ up to its d-th element is denoted by $\vec{s}_d = [s_0, s_1, \ldots, s_d]$. The formal definition of a signature is given in the Appendix, and we refer intrigued readers to Zakov et al. (2013) for an elaborated discussion on its properties.

We will use the notation $\vec{s} = [s_0, s_1, \ldots, s_d, 0, \ldots]$ to imply that the remaining signature elements after position d are all zeros. From property (2), it follows that for a signature $\vec{s}$ such that $\|\vec{s}\| = n$, all signature elements s_d for $d > \log n$ are zeros, thus signatures can be explicitly represented by a relatively small number of nonzero elements. In particular, from properties (1) and (2) it follows that the only signature of an empty collection is $\vec{s} = [0, 0, \ldots]$, and that the only signature of a collection containing a single element is $\vec{s} = [1, 0, \ldots]$. Otherwise, two collections of the same size may have different signatures. From property (3), wrapping an l-BFB palindrome collection (i.e., replacing each l-BFB palindrome β in the collection with an $(l-1)$-block $\sigma_{l-1}\beta\bar{\sigma}_{l-1}$) does not affect its signature.

Signatures can be ranked according to their lexicographic order. That is, say that $\vec{s} < \vec{s}'$ if there exists an index d such that $\vec{s}_{d-1} = \vec{s}'_{d-1}$ and $s_d < s'_d$, and say that $\vec{s} \leq \vec{s}'$ if $\vec{s} < \vec{s}'$ or $\vec{s} = \vec{s}'$. Lemma 2 below implies that the signature series $\vec{s}^{k+1}, \vec{s}^{k}, \ldots, \vec{s}^{1}$ corresponding to the block collections series $B^{k+1}, B^{k}, \ldots, B^{1}$ in a layered representation of a BFB palindrome is lexicographically nondecreasing.

Lemma 2 *Let B be an l-block collection with a signature $\vec{s}$. For any folding B' of B and its corresponding signature $\vec{s}'$, $\vec{s} \leq \vec{s}'$. In addition, for any signature $\vec{s}'$ such that (1) $\vec{s} \leq \vec{s}'$ and (2) $\vec{s}'$ is the lexicographically minimal signature among all signatures of cardinality $\|\vec{s}'\|$ that meet (1), there exists a folding B' of B whose signature is $\vec{s}'$.*

The proof of Lemma 2 follows from Claims 14 and 28 in Zakov et al. (2013) (Supporting Information). The signatures corresponding to the four block collections in Figure 3b are $\vec{s}^4 = [0, 0, 1, 0, \ldots]$, $\vec{s}^3 = [1, -1, 0, \ldots]$, $\vec{s}^2 = [1, 0, -1, 0, \ldots]$, and $\vec{s}^1 = [1, 0, \ldots]$, respectively. Observe that the cardinality of each signature equals the size of the corresponding collection (i.e., the corresponding count in $\vec{n} = [1, 5, 3, 4]$), and that $\vec{s}^{l+1} \leq \vec{s}^{l}$ for every $1 \leq l < 4$.

3.2. Valid signature series

Definition 3 *A* valid signature series *for a vector $\vec{n} = [n_1, n_2, \ldots, n_k]$ is a series of lexicographically nonincreasing signatures $\vec{s}^1, \vec{s}^2, \ldots, \vec{s}^k$, satisfying $\|\vec{s}^l\| = n_l$ for every $1 \leq l \leq k$, $\vec{s}^1 \leq [1, 0, \ldots]$, and $[0, 0, \ldots] \leq \vec{s}^k$.*

For convenience, we will sometimes consider the signatures $[1, 0, \ldots]$ and $[0, 0, \ldots]$ as fixed sentinel additions at the beginning and ending of a valid signature series and mark them respectively by $\vec{s}^0$ and $\vec{s}^{k+1}$. If $\vec{n}$ is a BFB count vector, there exists a BFB palindrome β with $\vec{n}(\beta) = 2\vec{n}$, and a corresponding collection series $B^0 = \{\beta\}, B^1, \ldots, B^k, B^{k+1} = \emptyset$ in the layers representation of β. Lemma 2 implies that the corresponding signature series for this collection series is a valid signature series for $\vec{n}$, since for every $0 \leq l \leq k$, B^l is obtained by applying a folding operation over B^{l+1} that can only increase the lexicographic signature rank, and a wrapping operation that does not change the signature. On the other hand, if there exists a valid signature series $\vec{s}^0 = [1, 0, \ldots], \vec{s}^1, \ldots, \vec{s}^k, \vec{s}^{k+1} = [0, 0, \ldots]$ for $\vec{n}$, it is possible to generate a BFB palindrome β with $\vec{n}(\beta) = 2\vec{n}$ as follows. Run the schematic layered algorithm described above with respect to $\vec{n}$, where each time a folding operation is applied it is such that yields the minimal signature increment with respect to its input and output collections. Since $\vec{s}(B^{k+1}) = \vec{s}^{k+1} = [0, 0, \ldots] \leq \vec{s}^k$ and $\|\vec{s}^k\| = n_k$, Lemma 2 implies that it is possible to fold B^{k+1} into a $(k+1)$-BFB palindrome collection of size n_k, and that the lexicographically minimal signature $\vec{s}'^k$ of such a collection satisfies $\vec{s}^{k+1} \leq \vec{s}'^k \leq \vec{s}^k \leq \vec{s}^{k-1}$. Inductively, each generated block collection B^l in this process has a corresponding signature $\vec{s}'^l$ that satisfies $\vec{s}'^l \leq \vec{s}^l \leq \vec{s}^{l-1}$, and from Lemma 2 it can be folded into the next collection in the series B^{l-1} without a folding failure. We hence get the following conclusion:

Conclusion 4 *A vector $\vec{n}$ is a BFB vector if and only if it has a valid signature series. Moreover, any subsequence of a BFB vector is also a BFB vector, evident by the corresponding subseries of a valid signature series for the full vector.*

For example, the vector $\vec{n}=[3,4]$ is a BFB vector, due to the valid signature series $\vec{s}^1=[1,-1,0,\ldots]$, $\vec{s}^2=[0,0,1,0,\ldots]$. A corresponding BFB string may be obtained by $\mathrm{AB} \overset{\mathrm{BFB}}{\longrightarrow} \mathrm{AB}\bar{\mathbf{B}} \overset{\mathrm{BFB}}{\longrightarrow} \mathrm{AB}\bar{\mathrm{B}}\mathbf{B}\bar{\mathbf{B}}\bar{\mathbf{A}} \overset{\mathrm{BFB}}{\longrightarrow} \mathrm{AB}\bar{\mathrm{B}}\mathrm{B}\bar{\mathrm{B}}\bar{\mathrm{A}}\mathbf{A}$. An example for a vector that does not have a valid signature series is the vector $\vec{n}=[4,3]$. The only signatures with cardinality 4 that rank lexicographically between $[0,0,\ldots]$ and $[1,0,\ldots]$ are the signatures $[0,0,1,0,\ldots]$ and $[0,2,0,\ldots]$, and the only such signature with cardinality 3 is $[1,-1,0,\ldots]$. The latter signature does not precede lexicographically any of the two possible 4-cardinality signatures, therefore no valid signature series for $\vec{n}$ exists.

A valid signature series for a given count vector, if exists, can be computed iteratively by processing the counts in the vector one by one. This process can be done either by traversing the counts from n_1 to n_k, or traversing them in a reversed order. Next, we describe this computation.

Let $\vec{n}=[n_1,n_2,\ldots,n_k]$ be a BFB vector, and let $1 \le l \le k+1$. Define the *right-maximal signature* $R(\vec{n}_{1,l-1})$ of the prefix $\vec{n}_{1,l-1}=[n_1,n_2,\ldots,n_{l-1}]$ of $\vec{n}$ to be $[1,0,\ldots]$ if $l=1$, and otherwise to be the lexicographically maximal signature $\vec{s}^{l-1}$ in some valid signature series $\vec{s}^1,\ldots,\vec{s}^{l-1}$ for $\vec{n}_{1,l-1}$. Similarly, define the *left-minimal signature* $L(\vec{n}_{l,k})$ of the suffix $\vec{n}_{l,k}=[n_l,n_{l+1},\ldots,n_k]$ of $\vec{n}$ to be $[0,0,\ldots]$ if $l=k+1$, and otherwise to be the lexicographically minimal signature $\vec{s}^l$ in some valid signature series $\vec{s}^l,\ldots,\vec{s}^k$ for $\vec{n}_{l,k}$.

Lemma 5 *Let $\vec{n}=[n_1,n_2,\ldots,n_k]$ be a BFB vector. For every $1 \le l' \le l \le k+1$, $L(\vec{n}_{l,k}) \le R(\vec{n}_{1,l-1})$, $R(\vec{n}_{1,l-1}) \le R(\vec{n}_{1,l'-1})$, and $L(\vec{n}_{l,k}) \le L(\vec{n}_{l',k})$.*

Proof: We start by showing the first inequality in the lemma. If $l=1$ or $l=k+1$, $L(\vec{n}_{l,k}) \le R(\vec{n}_{1,l-1})$ follows immediately. Otherwise, consider a valid signature series $\vec{s}^1,\vec{s}^2,\ldots,\vec{s}^k$ for $\vec{n}$. Note that its prefix $\vec{s}^1,\vec{s}^2,\ldots,\vec{s}^{l-1}$ is a valid signature series for $\vec{n}_{1,l-1}$, and its suffix $\vec{s}^l,\vec{s}^{l+1},\ldots,\vec{s}^k$ is a valid signature series for $\vec{n}_{l,k}$. Thus, by definition, $L(\vec{n}_{l,k}) \le \vec{s}^l \le \vec{s}^{l-1} \le R(\vec{n}_{1,l-1})$.

To show the second inequality in the lemma, let $\vec{s}^1,\vec{s}^2,\ldots,\vec{s}^{l-1}$ be a valid signature series for $\vec{n}_{1,l-1}$ such that $\vec{s}^{l-1}=R(\vec{n}_{1,l-1})$. Observe similarly as above that $R(\vec{n}_{1,l-1})=\vec{s}^{l-1} \le \vec{s}^{l'-1} \le R(\vec{n}_{1,l'-1})$. The last inequality in the lemma is shown symmetrically. ■

The MIN-DECREMENT procedure (Algorithm 1) gets as an input a signature $[0,0,\ldots] \le \vec{s}$ and an integer $n \ge 0$, and returns the lexicographically maximal signature $\vec{s}'$ such that $[0,0,\ldots] \le \vec{s}' \le \vec{s}$ and $\|\vec{s}'\|=n$ if such a signature exists, and otherwise it returns a fail message. Here, for an integer $m \ne 0$, the notation d_m represents the maximum integer such that m divides by 2^{d_m}. Thus, for example, $d_{13}=d_{13\cdot 2^0}=0$, and $d_{-12}=d_{-3\cdot 2^2}=2$. The correctness of this computation is shown in the Appendix. Symmetrically, the MIN-INCREMENT procedure gets as an input a signature $\vec{s} \le [1,0,\ldots]$ and an integer $n \ge 0$, and returns the lexicographically minimal signature $\vec{s}'$ such that $\vec{s} \le \vec{s}' \le [1,0,\ldots]$ and $\|\vec{s}'\|=n$ if such a signature exists, and otherwise it returns a fail message. The pseudocode for this procedure is given in the Appendix, and its proof is symmetric to that of the MIN-DECREMENT procedure.

Algorithm 1: MIN-DECREMENT($\vec{s}$, n)

```
    Input: A signature $[0,0,\ldots] \le \vec{s}$ and an integer $n \ge 0$.
    Output: The lexicographically maximal signature $[0,0,\ldots] \le \vec{s}' \le \vec{s}$ such that $\|\vec{s}'\|=n$, or the message
        "FAILED" if there is no such signature.
1   Let $m=\|\vec{s}\|-n$. If $m=0$ then return $\vec{s}$.
2
3   Else if there is an integer $0 \le d \le d_m$ such that $n \ge \|\vec{s}_{d-1}\|+2^d \max\{-s_d+1,0\}$ then
4       Let $d$ be the maximum integer meeting the condition above. Initialize $\vec{s}'$ so that
        $\vec{s}'_{d-1}=\vec{s}_{d-1}$, and $s'_d=s_d-2$ if $d<d_m$, or $s'_d=s_d-1$ if $d=d_m$.
5       If $n \ge \|\vec{s}'_d\|$ then set $s'_{d+1} \leftarrow \frac{n-\|\vec{s}'_d\|}{2^{d+1}}$.
6
7       Else set $s'_d \leftarrow \frac{n-\|\vec{s}'_{d-1}\|}{2^d}$.
8
9       If $[0,0,\ldots] \le \vec{s}'$ then return $\vec{s}'$,
10      else return "FAILED".
11  Else return "FAILED".
```

Lemma 6 *If $\vec{n}_{1,l-1}=[n_1, \ldots, n_{l-1}]$ is a BFB vector, $\vec{s}=R(\vec{n}_{1,l-1})$, and MIN-DECREMENT($\vec{s}, n_l$) does not fail and returns a signature $\vec{s}'$, then $\vec{s}'$ is the right-maximal signature for the BFB vector $\vec{n}_{1,l}=[n_1, \ldots, n_{l-1}, n_l]$. Symmetrically, if $\vec{n}_{l+1,k}=[n_{l+1}, \ldots, n_k]$ is a BFB vector, $\vec{s}=L(\vec{n}_{l+1,k})$, and MIN-INCREMENT($\vec{s}, n_l$) does not fail and returns a signature $\vec{s}'$, then $\vec{s}'$ is the left-minimal signature for the BFB vector $\vec{n}_{l,k}=[n_l, n_{l+1}, \ldots, n_k]$.*

Proof: We show the first part of the lemma, where the second part is shown symmetrically. First, note that the constructed vector $\vec{n}_{1,l}$ is indeed a BFB vector, due to the corresponding valid signature series obtained by adding $\vec{s}'$ to a valid signature series for $\vec{n}_{l-1}$ whose last signature is $\vec{s}$. Note that $\|R(\vec{n}_{1,l})\|=\|\vec{s}'\|=n_l$. From Lemma 5 $R(\vec{n}_{1,l})\leq\vec{s}$, and since $\vec{s}'=$MIN-DECREMENT($\vec{s}, n_l$) it follows that $R(\vec{n}_{1,l})\leq\vec{s}'$. From the maximality of $R(\vec{n}_{1,l})$, $R(\vec{n}_{1,l})=\vec{s}'$. ■

Lemma 6 implies a simple algorithm for deciding if a given vector $\vec{n}$ is a BFB vector. Such an algorithm tries generating a valid signature series for $\vec{n}$ either by starting from the first signature $\vec{s}^0=[1, 0, \ldots]$, and generating each signature $\vec{s}^l$ by applying the MIN-INCREMENT procedure with respect to $\vec{s}^{l-1}$ and n_l, or by starting from the last signature $\vec{s}^{k+1}=[0, 0, \ldots]$, and generating each signature $\vec{s}^l$ by applying the MIN-DECREMENT procedure with respect to $\vec{s}^{l+1}$ and n_l. If the MIN-INCREMENT or MIN-DECREMENT procedures fail at some stage, then the algorithm reports $\vec{n}$ not to be a BFB vector. Otherwise, the algorithm succeeds to generate a valid signature series for $\vec{n}$, and reports $\vec{n}$ to be a BFB vector. As a matter of fact, Algorithm DECISION-BFB in Zakov et al. (2013) is equivalent to the right-to-left version of the above algorithm.

3.3. Solving the exhaustive BFB variants

In this section, let W be a weight function, and $0 < \eta \leq 1$ some weight threshold. Let $0 \leq l \leq k$, and consider the set of all signature-weight pairs of the form $\langle R(\vec{n}_{1,l}), W(\vec{n}_{1,l})\rangle$ such that $\vec{n}_l=[n_1, n_2, \ldots, n_l]$ is a BFB vector and $W(\vec{n}_{1,l})\geq\eta$. Say that the pair $\langle\vec{s}, w\rangle$ within this set *dominates* the pair $\langle\vec{s}', w'\rangle$ if $\vec{s}'\leq\vec{s}$ and $w'\leq w$. Define the *l-th boundary curve* C^l with respect to W and η as the maximal subset of these pairs satisfying that no pair in C^l dominates another pair in C^l, and note that C^l is unique. Traversing the pairs in C^l from lowest to highest lexicographic signature rank, the series of signature values strictly increases, while the series of weight values strictly decreases, yielding a steplike curve (Fig. 4). Given W and η, Algorithm 2 generates boundary curves C^l for every $0 \leq l \leq k$, which will later be exploited by algorithms for the BFB exhaustive vector and string search variants.

Proof: [Algorithm 2] Note that a pair in C^0 corresponds to a right-maximal signature and a weight of an empty vector. By definition, the only such pair is the pair $\langle[1, 0, \ldots], 1\rangle$, and the algorithm correctly sets C^0 to contain this single pair (line 1). Now, assuming inductively the algorithm has computed correctly the curve C^{l-1}, we prove it also computes correctly C^l. It is clear from lines 6 and 7 of the algorithm that no pair in the set C^l computed by the algorithm dominates another pair in this set. It therefore remains to show that after the l-th loop iteration was executed: (1) for every BFB vector $\vec{n}_{1,l}=[n_1, \ldots, n_l]$ with $W(\vec{n}_{1,l})\geq\eta$ there exists a pair $\langle\vec{s}, w\rangle \in C^l$, which dominates $\langle R(\vec{n}_{1,l}), W(\vec{n}_{1,l})\rangle$, and (2) for every pair $\langle\vec{s}, w\rangle \in C^l$ there exists some BFB vector $\vec{n}_{1,l}=[n_1, \ldots, n_l]$ such that $\vec{s}=R(\vec{n}_{1,l})$ and $w=W(\vec{n}_{1,l})$.

Algorithm 2: BOUNDARY-CURVES (W, η)

```
    Input: A weight function $W$ and a weight $\eta$.
    Output: Boundary curves $C^l$ for every $0 \leq l \leq k$ with respect to $W$ and $\eta$.
1   Set $C^0 \leftarrow \{\langle[1, 0, \ldots], 1\rangle\}$.
2   For $l \leftarrow 1$ to $k$ do
3       Set $C^l \leftarrow \emptyset$.
4       For each $n$ and $\langle\vec{s}', w'\rangle \in C^{l-1}$ s.t. $w' \cdot w_{l,n} \geq \eta$ and MIN-DECREMENT ($\vec{s}', n$) does not fail do
5           Let $\vec{s}$ be the output of MIN-DECREMENT ($\vec{s}', n$), and let $w=w' \cdot w_{l,n}$.
6           If $\langle\vec{s}, w\rangle$ is not dominated by any pair in $C^l$ then
7               Add $\langle\vec{s}, w\rangle$ into $C^l$, and remove from $C^l$ all pairs dominated by $\langle\vec{s}, w\rangle$.
8   Return $\{C^0, C^1, \ldots, C^k\}$.
```

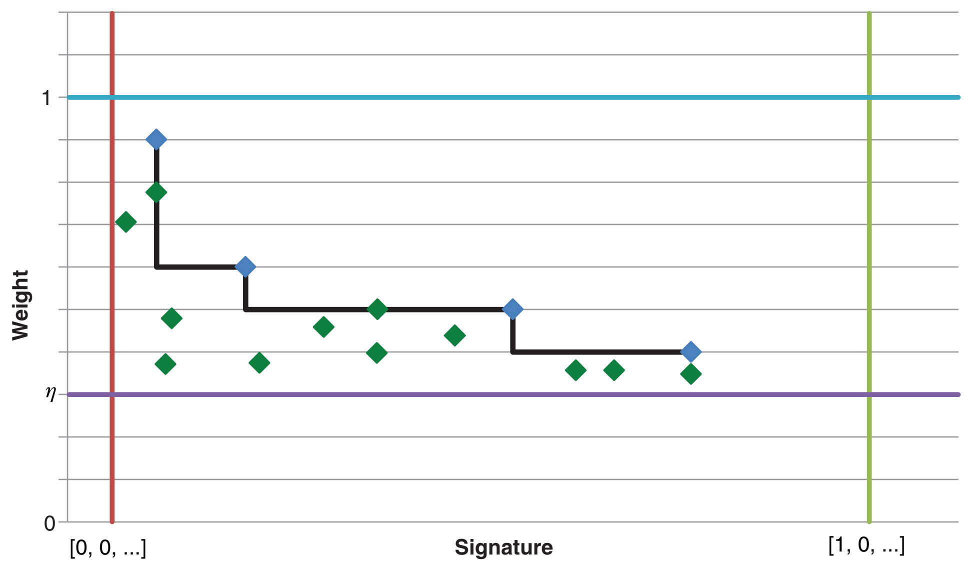

FIG. 4. A boundary curve. Points correspond to pairs of the form $\langle \vec{s}, w\rangle$, with x-coordinate reflecting the lexicographic rank of $\vec{s}$, and y-coordinate equaling w. Blue points belong to the boundary curve, and green points are dominated by points on the curve.

We start by showing (1). Let $\vec{n}_{1,l} = [n_1, \ldots, n_{l-1}, n_l]$ be a BFB vector with $W(\vec{n}_{1,l}) \geq \eta$, and consider its prefix $\vec{n}_{1,l-1} = [n_1, \ldots, n_{l-1}]$. Observe that $W(\vec{n}_{n,l-1}) = \frac{W(\vec{n}_{1,l})}{w_{l,n_l}} \geq \eta$. As $\vec{n}_{1,l-1}$ is also a BFB vector, the inductive assumption implies that C^{l-1} contains a pair $\langle \vec{s}', w'\rangle$ that dominates $\langle R(\vec{n}_{1,l-1}), W(\vec{n}_{1,l-1})\rangle$. From Lemma 5, $R(\vec{n}_{1,l}) \leq R(\vec{n}_{1,l-1}) \leq \vec{s}'$. Since $\|R(\vec{n}_{1,l})\| = n_l$, running MIN-DECREMENT $(\vec{s}', n_l)$ does not fail, and returns a signature $\vec{s}$ such that $R(\vec{n}_{1,l}) \leq \vec{s} \leq \vec{s}'$ and $\|\vec{s}\| = n_l$. As $w' \cdot w_{l,n_l} \geq W(\vec{n}_{1,l-1}) \cdot w_{l,n_l} = W(\vec{n}_{1,l}) \geq \eta$, it follows that the algorithm runs the code in lines 5–7 with respect to n_l and $\langle \vec{s}', w'\rangle$. In particular, the algorithm updates C^l with the pair $\langle \vec{s}, w\rangle$ for $w = w' \cdot w_{l,n_l} \geq W(\vec{n}_{1,l})$ (lines 6–7). Therefore, at the end of the l-th iteration, either C^l contains $\langle \vec{s}, w\rangle$, or it contains some other signature-weight pair that dominates $\langle \vec{s}, w\rangle$, and so it contains a pair that dominates $\langle R(\vec{n}_{1,l}), W(\vec{n}_{1,l})\rangle$.

To show (2), assume that C^l contains a pair $\langle \vec{s}, w\rangle$. This pair was added to C^l in line 7 of the algorithm, which means there exists some pair $\langle \vec{s}', w'\rangle \in C^{l-1}$ such that for $n_l = \|\vec{s}\|$, $\vec{s} = \text{MIN-DECREMENT}(\vec{s}', n_l)$, and $w = w' \cdot w_{l,n_l} \geq \eta$. From the inductive assumption, there is BFB vector $\vec{n}_{1,l-1} = [n_1, \ldots, n_{l-1}]$ such that $\vec{s}' = R(\vec{n}_{1,l-1})$ and $w' = W(\vec{n}_{1,l-1})$. For the vector $\vec{n}_{1,l} = [n_1, \ldots, n_{l-1}, n_l]$, lemma 6 implies that $\vec{s} = R(\vec{n}_{1,l})$. In addition, $W(\vec{n}_{1,l}) = w$, and the lemma follows. ■

In Appendix section 6.2 we show that the number a_n of all signatures with cardinality n satisfies $a_n = 2^{\Theta(\log^2 n)}$. Since no two pairs in a boundary curve share the same signature, the number of pairs in a boundary curve with cardinality n is bounded by $2^{O(\log^2 n)}$. It would be quite realistic to assume that for a segment l and its corresponding most likely count n_l^*, all counts n such that $w_{l,n} \geq \eta$ satisfy $n = O((n_l^*)^x)$ for some constant x. This implies that the total number of different elements in a boundary curve is bounded by the subexponential term $O((n_l^*)^x) \cdot 2^{O(\log^2 n_l^*)} = 2^{O(\log^2 n_l^*)}$. In particular, the number of candidates n, $\langle \vec{s}', w'\rangle$ examined in line 4 of Algorithm 2 is $O((n_l^*)^x) \cdot 2^{O(\log^2 n_{l-1}^*)}$. Over each such candidate, the condition in line 6 is examined, and it may induce at most one insertion of a pair into C^l, and possibly one future deletion from C^l (if the inserted pair is dominated by a pair that is inserted into C^l later on). It is possible to maintain the pairs in C^l sorted, and implement the condition check in line 6 and insertions and deletions from C^l in line 7 in $O(\log |C^l|) = O(\log^2 n_l^*)$ time per operation, for example, using a self-balancing binary search tree (Knuth, 1998). Thus, the l-th iteration of the algorithm runs in $O((n_l^*)^x(\log^2 n_l^*)) \cdot 2^{O(\log^2 n_{l-1}^*)} = 2^{O(\max\{\log n_l^*, \log^2 n_{l-1}^*\})}$ time, and the total running time of the algorithm is $\sum_{l=1}^{k} 2^{O(\log^2 n_l^*)}$.

Next, we present Algorithm 3 for the BFB exhaustive vector search variant. The algorithm processes the segments of the input one by one, starting from the k-th segment down to the first segment. The notation $[n, \vec{n}]$ is used for denoting a vector whose first element is the integer n, and its remaining suffix is the vector $\vec{n}$.

Proof: [Algorithm 3] By definition, if the boundary curve C^k is empty, it implies there is no BFB vector $\vec{n} = [n_1, \ldots, n_k]$ with $W(\vec{n}) \geq \eta$. In this case, the algorithm correctly reports there is no solution to the input (line 1).

Otherwise, we show for every $1 \leq l \leq k+1$ that the following invariant holds: *After Q^l is fully computed, Q^l contains $\vec{n}_{l,k} = [n_l, \ldots, n_k]$ if and only if $\vec{n}_{l,k}$ is a suffix of some BFB vector $\vec{n} = [n_1, \ldots, n_k]$ of weight $W(\vec{n}) \geq \eta$.* In particular, this invariant proves that the returned value Q^1 (line 9) is indeed the solution for the BFB exhaustive vector search variant, and so it only remains to establish the correctness of the invariant.

Algorithm 3: EXHAUSTIVE-VECTOR-SEARCH (W, η)

```
  Input: A weight function $W$ and a weight $0 < \eta \leq 1$.
  Output: All BFB vectors $\vec{n} = [n_1, n_2, \ldots, n_k]$ satisfying $W(\vec{n}) \geq \eta$.
1 Generate all boundary curves $C^0, C^1, \ldots, C^k$ with respect to $W$ and $\eta$ using Algorithm 2. If $C^k$ is empty,
  return the message ''NO SOLUTION'' and halt.
2 Set $Q^{k+1}$ to be the collection containing a single empty vector.
3 For $l \leftarrow k$ down to 1 do
4     Set $Q^l \leftarrow \emptyset$.
5     For each $\vec{n}_{l+1,k} \in Q^{l+1}$ and count $n$ such that $W([n, \vec{n}_{l+1,k}]) \geq \eta$ and MIN-INCREMENT
      $(L(\vec{n}_{l+1,k}), n)$ does not fail do
6         Let $\vec{n}_{l,k} = [n, \vec{n}_{l+1,k}]$, and let $\vec{s} =$ MIN-INCREMENT$(L(\vec{n}_{l+1,k}), n)$.
7         If there exists a pair $\langle \vec{s}', w' \rangle \in C^{l-1}$ such that $\vec{s} \leq \vec{s}'$ and $w' \cdot W(\vec{n}_{l,k}) \geq \eta$ then
8             Add $\vec{n}_{l,k}$ to $Q^l$.
9 Return $Q^1$.
```

For $l = k + 1$, the fact that Q^{k+1} contains a single empty suffix (line 2) derives the invariant in a straightforward manner. Otherwise, assuming inductively the invariant holds with respect to Q^{l+1}, we prove it also holds with respect to Q^l.

Let $\vec{n} = [n_1, \ldots, n_k]$ be a BFB vector of weight $W(\vec{n}) \geq \eta$, and consider its two suffixes $\vec{n}_{l,k} = [n_l, n_{l+1} \ldots, n_k]$ and $\vec{n}_{l+1,k} = [n_{l+1} \ldots, n_k]$. From the inductive assumption, $\vec{n}_{l+1,k} \in Q^{l+1}$. From Lemma 6, $\vec{s} = L(\vec{n}_{l,k})$ satisfies that $\vec{s} = \text{MIN-INCREMENT}(L(\vec{n}_{l+1,k}), n_l)$. Since $W(\vec{n}_{l,k}) \geq W(\vec{n}) \geq \eta$, the condition in line 5 holds, and lines 6–8 are executed with respect to $\vec{n}_{l,k}$ and $\vec{s}$. Note that the prefix $\vec{n}_{1,l-1} = [n_1, \ldots, n_{l-1}]$ of $\vec{n}$ is a BFB vector with $W(\vec{n}_{1,l-1}) \geq W(\vec{n}) \geq \eta$. From the definition of C^{l-1}, there exists a pair $\langle \vec{s}', w' \rangle \in C^{l-1}$ that dominates the pair $\langle R(\vec{n}_{1,l-1}), W(\vec{n}_{1,l-1}) \rangle$. From Lemma 5, $L(\vec{n}_{l,k}) \leq R(\vec{n}_{1,l-1}) \leq \vec{s}'$. In addition, $w' \cdot W(\vec{n}_{l,k}) \geq W(\vec{n}_{1,l-1}) \cdot W(\vec{n}_{l,k}) = W(\vec{n}) \geq \eta$, and so the condition in line 7 holds, and the algorithm adds $\vec{n}_{l,k}$ into Q^l in line 8.

For the other direction of the invariant, let $\vec{n}_{l,k} = [n_l, n_{l+1}, \ldots, n_k] \in Q^l$. Due to the manner it was constructed (lines 5–6), its suffix $\vec{n}_{l+1,k} = [n_{l+1}, \ldots, n_k]$ is in Q^{l+1}, and from Lemma 6, $\vec{n}_{l,k}$ is a BFB vector with $L(\vec{n}_{l,k}) = \vec{s}$. From line 7, there exists a pair $\langle \vec{s}', w' \rangle \in C^{l-1}$ such that $\vec{s} \leq \vec{s}'$ and $w' \cdot W(\vec{n}_{l,k}) \geq \eta$, and so from the definition of C^{l-1} there exists a BFB vector $\vec{n}_{1,l-1} = [n_1, \ldots, n_{l-1}]$ for which $R(\vec{n}_{1,l-1}) = \vec{s}'$ and $W(\vec{n}_{1,l-1}) = w'$. The concatenation of $\vec{n}_{1,l-1}$ and $\vec{n}_{l,k}$ gives the vector $\vec{n} = [n_1, \ldots, n_{l-1}, n_l, \ldots, n_k]$, whose weight satisfies $W(\vec{n}) = W(\vec{n}_{1,l-1}) \cdot W(\vec{n}_{l,k}) = w' \cdot W(\vec{n}_{l,k}) \geq \eta$. In addition, $\vec{n}$ is a BFB vector, due to the corresponding valid signature series obtained by concatenating a valid signature series for $\vec{n}_{1,l-1}$ that ends with $\vec{s}'$ and a valid signature series for $\vec{n}_{l,k}$ that starts with $\vec{s}$, concluding this direction of the proof. ■

Finally, we describe an algorithm for the exhaustive BFB string search variant. This algorithm applies a similar approach to the exhaustive vector search algorithm in order to produce all BFB strings whose count vector weights are at least η. The pseudocode for this computation is given in Algorithm 4. It starts by generating signature curves exactly as done by Algorithm 3. Then, in each iteration l, instead of computing a set Q^l of count vectors, the algorithm computes a set P^l of l-block collections. The initial collection P^{k+1} contains a single empty $(k + 1)$-block collection. In the l-th iteration, for each $(l + 1)$-block collection $B^{l+1} \in P^{l+1}$, all possible foldings of B^{l+1} are enumerated, and each such folding is rendered into an l-block collection B^l as described in section 3.1 (the term ''wrapped folding'' in line 5 of the algorithm refers to these l-block collections). The notation $W(B^l)$ is used for denoting the weight of the vector $\vec{n}_{l,k}$ such that $2\vec{n}_{l,k}$ is the summation of count vectors of all strings in B^l. The signature and weight of B^l are examined against C^{l-1} similarly as done in line 7 of Algorithm 3, and if meeting the condition B^l is added into P^l. After P^1 is computed, all foldings of collections B^1 in P^1 into 1-BFB palindromes are enumerated, and all half-length prefixes of such palindromes are reported. Algorithm 4 can be proven similarly to Algorithm 3, using the following invariant: At the end of the l-th iteration, P^l contains all l-block collections B^l such that there exists some 1-BFB palindrome β in which the l-th layer's block collection is B^l, and the weight of the vector $\vec{n}$ such that $\vec{n}(\beta)=2\vec{n}$ satisfies $W(\vec{n})\geq\eta$.

Algorithm 4: EXHAUSTIVE-STRING-SEARCH (W,η)

```
   Input: A weight function $W$ and a weight $0 < \eta \leq 1$.
   Output: All BFB strings $\alpha$ satisfying $W(\vec{n}(\alpha))\geq\eta$.
1  Generate all boundary curves $C^0, C^1, \ldots, C^k$ with respect to $W$ and $\eta$ using Algorithm 2. If $C^k$ is empty,
   return the message "NO SOLUTION" and halt.
2  Set $P^{k+1}$ to be the collection containing a single empty collection $B^{k+1}$.
3  For $l \leftarrow k$ down to 1 do
4      Set $P^l \leftarrow \emptyset$.
5      For each $B^{l+1} \in P^{l+1}$ and for every wrapped folding $B^l$ of $B^{l+1}$ do
6          If there exists a pair $\langle\vec{s}, w\rangle \in C^{l-1}$ such that $\vec{s}(B^l)\leq\vec{s}$ and $w \cdot W(B^l) \geq \eta$ then
7              Add $B^l$ to $P^l$.
8  For each $B^1 \in P^1$ and for every folding of $B^1$ into a collection with a single 1-BFB palindrome $\{\alpha\bar{\alpha}\}$ do
9      Report $\alpha$ to be a BFB string with $W(\vec{n}(\alpha))\geq\eta$.
```

4. RESULTS

In order to test our algorithms we have used cancer data taken from the Cancer Genome Project dataset (Bignell et al., 2010). This data covers aCGH samples (Affymetrix Genome-Wide Human SNP Array 6.0) from 746 human cancer cell lines. Segmentation and segment copy numbers are as reported by Bignell et al. (2010), who used the PICNIC software (Greenman et al., 2010) for this analysis. In total, the dataset contains about 35,000 chromosomal arms (746 samples, 23 or 24 chromosomes per sample, 2 arms per chromosome), each arm is segmented, and each segment is assigned an estimated copy number based on the observed aCGH data. As shown in Zakov et al. (2013), short BFB-like count vectors have a high probability to emerge even when the genome was rearranged with mechanisms different from BFB. Thus, in order to detect significant BFB evidence we have filtered the set of chromosomal arms to include only arms with at least eight consecutive segments such that no adjacent segments share the same copy number estimation. After this filtration, the remaining subset included 6589 chromosomal arms. As the estimated counts reflect the expected segment copy numbers in all copies of the chromosome in the sample, we have corrected the counts by reducing $p - 1$ from each count, where p is the ploidy (i.e., the number of copies) of the chromosome in the sample. Typically $p = 2$, but since these are heavily rearranged cancer genomes, chromosomal losses and whole chromosomal duplications are not rare. We therefore allowed the value of p to vary between 1 and 5, and run the BFB analyses for each value.

As currently no analysis tool available produces count weights, we have derived such weights from the expected counts reported by PICNIC (after correcting for ploidy). Specifically, for a segment whose observed count is n, the weight of a count n' was defined by $\frac{\Pr(n|n')}{\Pr(n|n)}$, where $\Pr(x|\lambda)=\frac{\lambda^x e^{-\lambda}}{x!}$ is the probability to observe the value x for a random variable distributing according to the Poisson distribution with parameter λ. For each of the obtained weight functions, we used the DISTANCE-BFB algorithm from Zakov et al. (2013) to report all longest BFB subvectors with weight at least $\eta = 0.7$. Out of the 6589 samples, 54 samples had for at least one ploidy value $1 \leq p \leq 5$ a BFB subvector of length at least 8. Some samples had long BFB subvectors with respect to more than one ploidy value, and the total number of obtained BFB vectors was 86.

Then, we considered the segment coordinates and weight functions corresponding to the obtained subvectors and ran Algorithm 3 in order to find all BFB vectors of weights at least $\eta = 0.7$ with respect to these weight functions. For these 86 instances, a total number of 19154 heavy BFB vectors were found, with an average of 222 solutions per instance. This reveals an interesting property of the problem when applied over this data: the vast majority of samples, 6535 out of 6589, cannot be explained by any BFB count vector (and thus are unlikely to be obtained from BFB), yet each one of those 54 samples that can be explained by BFB has about several tens or hundreds of corresponding count vectors.

The above analysis was run by two variants of our algorithm—the IS variant described by Algorithm 3, and a variant that runs a similar procedure without applying the IS optimization (essentially, it runs the same code as Algorithm 3, with the exceptions that it does not generate the boundary curves in line 1 and does not apply the condition in line 7 before adding new elements to collections Q^l). The disadvantage of the non-IS variant is in that sets of the form Q^l maintains BFB vectors $\vec{n}_{l,k} = [n_l, \ldots, n_k]$, which may not be suffixes of some BFB vectors $\vec{n} = [n_1, \ldots, n_k]$ of weight at least η. To measure the gain of the IS algorithm, we count the number of signature increment attempts the algorithms perform (line 5). On average, the IS variant performed 57-fold less increments, with a total number of 5672346 increment attempts over all 86 vectors, versus 325343441 for the non-IS algorithm. While the IS variant has a clear efficiency advantage over the non-IS variant, this advantage might be considered more modest than expected. A possible reason for that is that maximum copy number values reported in Bignell et al. (2010) were limited to 14, even when the data suggests higher copy numbers. In general, higher copy numbers usually imply a higher number of alternative heavy counts, which in turn induce a higher number of possible heavy count vectors. For example, when comparing the two algorithms over the synthetic count vector $\vec{n}$ = [3, 8, 111, 8, 5, 150, 11, 170, 4, 53, 100, 75, 49, 10, 42, 18], using the same Poisson-based weights as described above and requiring that output vectors weigh at least $\eta = 0.85$, the non-IS algorithm runs 218 seconds[1] and performs over 20 million signature increments, whereas the IS algorithm runs 120 milliseconds and performs 635 signature increments. Both algorithms return exactly the same output—a set of 18 BFB vectors. Other simulated inputs can cause memory explosion for the non-IS variant, while being handled efficiently by the IS variant.

5. DISCUSSION AND CONCLUSIONS

The problem of detecting breakage fusion bridge is challenging, but significant progress has been made in the last few years. Our work suggests that while rare, BFB does occur in tumor-derived cell lines and also in primary tumors. In this work, we describe algorithms that can be used to enumerate all possible BFB architectures given uncertain copy number data.

The results of our analyses heavily depend on the input weights, which in turn depend on separated analyses applied to biological data. While we used here a simple Poisson-based model in order to render fixed available count estimations into weight functions, it is clear that more realistic weighing can be applied. Examining Figure 2, for example, one can observe that different segments demonstrate different variance in signal intensities, implying that some count estimates are more reliable than others. Incorporating segment lengths and signal variance information when choosing count weights is likely to produce more meaningful weights and improve the quality of the analyses output.

Different measurements can yield other types of BFB evidence. For example, deep sequencing experiments can sequence reads spanning genomic breakpoints. In a BFB modified genome, it is expected that many of these breakpoints reflect fold-back inversions (i.e., concatenations between reference segments and their inverted form), while such fold-back patterns are less common in other rearrangement mechanisms (Campbell et al., 2010). Thus, identification of high or low fold-back pattern frequencies can support or weaken the conjecture BFB has occurred, respectively. Such evidence is less frequent in currently available data, as reliable breakpoint information requires sequencing to a relatively high depth of coverage (while copy number data can be obtained also from sequencing with a lower depth of coverage or from aCGH experiments). When given though, such information can be integrated and improve the quality of BFB calling (Zakov et al., 2013).

As a last note, we would like to point out the fact that the concept of Informed Search (IS) was used here in a slightly unorthodox manner. Generally, IS methods attempt to reduce computation time in practice by exploiting additional information about the search space (given as an input to the algorithm or such that can be efficiently computed). Typically, such methods apply heuristic information for prioritizing the order in which different regions of the search space are examined to accelerate the search for a single solution to the input instance (e.g., the A* and AO* algorithms in Pearl, 1984). In contrast, the two search algorithms

[1]Running time was measured for an intel Core i7 processor with Microsoft Windows 7 operating system; code is implemented in Java.

described in this article exploit exact information encapsulated in the computed boundary curves, utilize it for pruning the search space from regions that are guaranteed to contain no solution to the given instance, and thus accelerate the search for all solutions.

6. APPENDIX

In this Appendix we complete some of the technical details omitted above. We show how BFB palindromes are composed recursively from shorter palindromes, describe how to derive signatures of BFB palindrome collections, show there are $2^{\Theta(\log^2 n)}$ different signatures of cardinality n, prove the MIN-DECREMENT algorithm correctness, and give the pseudocode for the MIN-INCREMENT algorithm. Most of the material in this section appears in Zakov et al. (2013) and is given here for completeness, except for the lower bound over the number of signatures with a given cardinality, which is first established here.

6.1. *Recursive decomposition of BFB palindromes*

Definition 7 *A string α is a* convexed l-palindrome *if $\alpha = \varepsilon$, or $\alpha = \gamma\beta\gamma$, γ is a convexed l-palindrome, β is an l-BFB palindrome, and top(γ) < top(β).*

Thus, for example, the following strings are all convexed 1-palindromes: $\gamma = \varepsilon\mathrm{A\bar{A}}\varepsilon$ [a 1-block with top(γ) = 1], $\gamma' = \mathrm{A\bar{A}}\mathbf{AB\bar{B}\bar{A}}\mathrm{A\bar{A}} = \gamma\beta\gamma$ [for the 1-bock $\beta = \mathrm{AB\bar{B}\bar{A}}$, with top($\gamma'$) = top($\beta$) = 2], and $\gamma'' = \mathrm{A\bar{A}AB\bar{B}\bar{A}A\bar{A}}\mathbf{ABC\bar{C}\bar{B}BC\bar{C}\bar{B}\bar{A}}\mathrm{A\bar{A}AB\bar{B}\bar{A}A\bar{A}} = \gamma'\beta'\gamma'$ [for the 1-BFB palindrome $\beta' = \mathrm{ABC\bar{C}\bar{B}BC\bar{C}\bar{B}\bar{A}}$, with top($\gamma''$) = top($\beta'$) = 3]. Note that every l-BFB palindrome α is also a convexed l-palindrome, since either $\alpha = \varepsilon$ or $\alpha = \varepsilon\alpha\varepsilon$.

Claim 1 in Zakov et al. (2013) *A string α is an l-BFB palindrome if and only if $\alpha = \varepsilon$, α is an l-block, or $\alpha = \beta\gamma\beta$, such that β is an l-BFB palindrome, γ is a convexed l-palindrome, and top(γ) $\leq$ top(β).*

Therefore, for the 1-BFB palindrome $\beta = \mathrm{AB\bar{B}\bar{A}}$ and the convexed 1-palindrome $\gamma = \mathrm{A\bar{A}AB\bar{B}\bar{A}A\bar{A}}$, the string $\alpha = \beta\gamma\beta = \mathrm{AB\bar{B}\bar{A}}\mathbf{A\bar{A}AB\bar{B}\bar{A}A\bar{A}}\mathrm{AB\bar{B}\bar{A}}$ is a 1-BFB palindrome. A BFB process that yields this string can be, for example, $\alpha_{1,2} = \mathrm{AB} \overset{\text{BFB}}{\longrightarrow} \mathrm{AB}\mathbf{\bar{B}}\mathrm{\bar{A}} \overset{\text{BFB}}{\longrightarrow} \mathrm{AB\bar{B}\bar{A}}\mathbf{A} \overset{\text{BFB}}{\longrightarrow} \mathrm{AB\bar{B}\bar{A}A}\mathbf{\bar{A}AB} \overset{\text{BFB}}{\longrightarrow} \mathrm{AB\bar{B}\bar{A}A\bar{A}AB}\mathbf{\bar{B}\bar{A}A\bar{A}AB\bar{B}\bar{A}}$. More generally, the above claim lays the rules for constructing BFB palindromes by concatenating shorter BFB palindromes and convexed palindromes, rather than applying a sequence of BFB cycles. Its proof is given in Zakov et al. (2013). These composition rules are used in order to enumerate all foldings of a given l-BFB palindrome collection by the exhaustive BFB string search algorithm.

6.2. *Signature computation and counting*

Let $B = \{m_1\beta_1, m_2\beta_2, \ldots, m_q\beta_q\}$ be an l-BFB palindrome collection. Define mod2 (B) to be the subcollection of B containing a single copy of each distinct element with an odd count in B. For example, for B = $\{2\beta_1, \beta_2, 5\beta_3, 6\beta_4\}$, mod2 ($B$) = $\{\beta_2, \beta_3\}$. Define $\frac{B}{2} = \{\lfloor\frac{m_1}{2}\rfloor\beta_1, \lfloor\frac{m_2}{2}\rfloor\beta_2, \ldots, \lfloor\frac{m_q}{2}\rfloor\beta_q\}$. In the above example, $\frac{B}{2} = \{\beta_1, 2\beta_3, 3\beta_4\}$. Observe that $B = \text{mod2}(B) + 2(\frac{B}{2})$.

In order to compute the signature of B, we first recursively decompose it into subcollections. Define $B_0 = B$. For every $d \geq 0$, define L_d = mod2 (B_d), $t_d = \min_{\beta\in L_d}$ top(β) or $t_d = \infty$ when $L_d = \emptyset$, $H_d = \{\beta \in (B_d - L_d) : \text{top}(\beta) \geq t_d\}$, and $B_{d+1} = \frac{B_d - L_d - H_d}{2}$. Now, the signature $\vec{s} = \vec{s}(B) = [s_0, s_1, \ldots]$ is computed as follows: $s_0 = |L_0|$, and $s_d = |L_d| - |L_{d-1}| - \frac{|H_{d-1}|}{2} + \max\{s_{d-1}, 0\}$ for every $d > 0$. Table 1 gives a signature computation example for a collection $B = \{2\beta_1, 5\beta_2, 6\beta_3, 2\beta_4, 4\beta_5\}$. We assume that elements are ordered with decreasing top values, that is top(β_i) $\geq$ top(β_{i+1}) for i = 1,2,3,4. It can be asserted that the signature cardinality equals to the collection size: $\|\vec{s}(B)\| = \sum_{d=0}^{\infty} 2^d \text{abs}(s_d) = 1 \cdot 1 + 2 \cdot 1 + 4 \cdot 2 + 8 \cdot 1 = 19 = |B|$.

Next, we show how to count the number of different signatures with a given cardinality. The only signature with cardinality 0 is the signature $[0, 0, \ldots]$. In a signature $\vec{s}$ with cardinality $\|\vec{s}\| > 0$ there must be at least one nonzero element. It can be asserted from the above signature definition that the first nonzero element in a signature must be positive. Nevertheless, we will relax this requirement and assume a signature can be any series of integers. Let b_n denote the number of such relaxed signatures of cardinality n, and let a_n denote the number of signatures of cardinality n in which the first nonzero element is positive. The only signatures with cardinality 1 are the signatures $[1, 0, \ldots]$ and $[-1, 0, \ldots]$. Therefore, $a_0 = a_1 = 1$

TABLE 1. SIGNATURE COMPUTATION

d	B_d	L_d	H_d	s_d
0	$\{2\beta_1, 5\beta_2, 6\beta_3, 2\beta_4, 4\beta_5\}$	$\{\beta_2\}$	$\{2\beta_1, 4\beta_2\}$	1
1	$\{3\beta_3, \beta_4, 2\beta_5\}$	$\{\beta_3, \beta_4\}$	$\{2\beta_3\}$	-1
2	$\{\beta_5\}$	$\{\beta_5\}$	$\emptyset$	-2
3	$\emptyset$	$\emptyset$	$\emptyset$	-1
4	$\emptyset$	$\emptyset$	$\emptyset$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

and $b_0 = 1, b_1 = 2$. With the exception of $n = 0$, it is simple to observe that $a_n = \frac{b_n}{2}$, for example, by observing that any signature $\vec{s}$ of the latter kind can be uniquely matched to a pair of signatures of the former kind—$\vec{s}$ itself, and the signature $\vec{s}'$ in which all elements have the same absolute values as in $\vec{s}$ and opposite signs.

For $n > 1$, partition the set of all signatures $\vec{s} = [s_0, s_1, \ldots]$ with $\|\vec{s}\| = n$ into two subsets: signatures with $\text{abs}(s_0) > 1$, and signatures with $\text{abs}(s_0) \leq 1$. Every signature $\vec{s}$ in the first group corresponds to a unique signature $\vec{s}'$ of cardinality $n - 1$, where $s'_0 = s_0 - 1$ if $s_0 > 1$ and $s'_0 = s_0 + 1$ if $s_0 < -1$, and all other elements in $\vec{s}'$ equal to the corresponding elements in $\vec{s}$. Every signature $\vec{s}$ in the second group corresponds to a unique signature $\vec{s}'$ of cardinality $\lfloor \frac{n}{2} \rfloor$, which is obtained by removing from $\vec{s}$ its first element (i.e., setting $s'_d = s_{d+1}$ for every $d \geq 0$). Therefore, the sizes of these groups are b_{n-1} and $b_{\lfloor \frac{n}{2} \rfloor}$, respectively, and so $b_n = b_{n-1} + b_{\lfloor \frac{n}{2} \rfloor}$. As $a_n = \frac{b_n}{2}$, we get that

$$a_n = a_{n-1} + a_{\lfloor \frac{n}{2} \rfloor}, \tag{1}$$

with the initial terms $a_0 = a_1 = 1$. The series $\{a_n\}$ is cataloged in the On-Line Encyclopedia of Integer Sequences (OEIS), entry A033485 (Sloane, 2007). Next, we show subexponential lower and upper bounds over a_n.

Claim 8 *The series $\{a_n\}$ satisfies $a_n = 2^{\Theta(\log^2 n)}$ for every integer $n \geq 0$.*

Proof: To show an upper bound, observe that for $\vec{s} = [s_0, s_1, \ldots]$ with $\|\vec{s}'\| = n$ it immediately follows that $\text{abs}(s_d) \leq \frac{n}{2^d}$ for every $d \geq 0$, and in particular $s_d = 0$ for $d > \log n$. For $d \leq \log n$, $-\frac{n}{2^d} \leq s_d \leq \frac{n}{2^d}$, and so it is possible to represent s_d using $\log n - d + 2$ bits. Thus, the number of bits needed for representing $\vec{s}$ can be bounded by $\sum_{0 \leq d \leq \log n} (\log n - d + 2) \leq \log^2 n$, and the total number of different signatures with cardinality n is at most $2^{\log^2 n}$.

A lower bound over a_n is next given by showing that $a_n \geq 2^{\frac{1}{4}\log^2 n}$. For $n < 4$ the inequality can be asserted in a simple manner. For $n \geq 4$, assuming inductively that $a_{n'} \geq 2^{\frac{1}{4}\log^2 n'}$ for every $n' < n$, we show that $a_n \geq 2^{\frac{1}{4}\log^2 n}$. First, observe that $a_n = a_{n-1} + a_{\lfloor \frac{n}{2} \rfloor} = a_{n-2} + a_{\lfloor \frac{n-2}{2} \rfloor} + a_{\lfloor \frac{n}{2} \rfloor} \geq a_{n-2} + 2a_{\lfloor \frac{n-2}{2} \rfloor}$. When n divides by 4,

$$\begin{aligned}
a_n &\geq a_{n-2} + 2a_{\frac{n-2}{2}} \\
&\geq a_{n-4} + 2a_{\frac{n-4}{2}} + 2a_{\frac{n-2}{2}} \geq a_{n-4} + 4a_{\frac{n-4}{2}} \\
&\geq a_{n-6} + 2a_{\frac{n-6}{2}} + 4a_{\frac{n-4}{2}} \geq a_{n-6} + 6a_{\frac{n-6}{2}} \\
&\vdots \\
&\geq a_{\frac{n}{2}} + \frac{n}{2} \cdot a_{\frac{n}{4}}
\end{aligned}$$

From the inductive assumption, we get that $a_n \geq \frac{n}{2} \cdot a_{\frac{n}{4}} \geq 2^{\log(n)-1} \cdot 2^{\frac{1}{4}(\log^2(\frac{n}{4}))} = 2^{\log(n)-1+\frac{1}{4}(\log(n)-\log(4))^2} = 2^{\log(n)-1+\frac{1}{4}(\log^2(n)-4\log(n)+4)} = 2^{\frac{1}{4}\log^2(n)}$. When n divides by 4 with a reminder of 1, 2, or 3, the inequality is proven similarly. ■

6.3. The MIN-INCREMENT and MIN-DECREMENT procedures

We next turn to prove the correctness of the MIN-DECREMENT procedure, and give the pseudocode of the symmetric MIN-INCREMENT procedure. We start by showing in Lemma 9 an elementary property of signatures, and then use this property to derive the procedures' implementation.

Lemma 9 *Let $\vec{s}$ and $\vec{s}'$ be two signatures, such that $\|\vec{s}\| - \|\vec{s}'\| = m \neq 0$. Then, there exists an index $0 \leq d \leq d_m$ such that $s_d \neq s'_d$. In addition, for the minimum such index d, $s_d - s'_d$ is even when $d < d_m$, and $s_d - s'_d$ is odd when $d = d_m$.*

Proof: By definition, m does not divide by 2^{d_m+1}, and so

$$\begin{aligned} m \text{ modulo } 2^{d_m+1} &= (\|\vec{s}\| - \|\vec{s}'\|) \text{ modulo } 2^{d_m+1} \\ &= \left(\sum_{i=0}^{\infty} 2^i \text{abs}(s_i) - \sum_{i=0}^{\infty} 2^i \text{abs}(s'_i)\right) \text{ modulo } 2^{d_m+1} \\ &= \left(\sum_{i=0}^{d_m} 2^i \text{abs}(s_i) - \sum_{i=0}^{d_m} 2^i \text{abs}(s'_i)\right) \text{ modulo } 2^{d_m+1} \\ &= (\|\vec{s}_{d_m}\| - \|\vec{s}'_{d_m}\|) \text{ modulo } 2^{d_m+1} \neq 0, \end{aligned}$$

therefore, there must be an index $0 \leq d \leq d_m$ such that $s_d \neq s'_d$. Let d be the minimum such index (where $\vec{s}_{d-1} = \vec{s}'_{d-1}$). Similarly as above,

$$\begin{aligned} m \text{ modulo } 2^{d+1} &= (\|\vec{s}_d\| - \|\vec{s}'_d\|) \text{ modulo } 2^{d+1} \\ &= (2^d(\text{abs}(s_d) - \text{abs}(s'_d))) \text{ modulo } 2^{d+1}. \end{aligned}$$

Now, if $d < d_m$ then m modulo $2^{d+1} = 0$, therefore abs(s_d) − abs(s'_d) must be even, and in particular $s_d - s'_d$ is even. If $d = d_m$, m modulo $2^{d+1} \neq 0$, abs(s_d) − abs(s'_d) must be odd, and in particular $s_d - s'_d$ is odd. Next, we show the correctness of the MIN-DECREMENT procedure (Algorithm 1). ■

Proof: [MIN-DECREMENT] First, note that if $\|\vec{s}\| = n$, then $\vec{s}$ satisfies the requirements on the output of the procedure, and due to line 1 in the procedure $\vec{s}$ is indeed the returned signature.

Otherwise, assume there exists a signature $\vec{s}^*$ such that $\vec{s}^* \leq \vec{s}$, $\|\vec{s}^*\| = n$, and $\vec{s}^*$ is the lexicographically maximal signature among all signatures satisfying these requirements. For this purpose, we do not make the assumption that $[0, 0, \ldots] \leq \vec{s}^*$. We will show that if $\vec{s}^*$ exists then the signature $\vec{s}'$ computed in lines 3–5 of the procedure equals to $\vec{s}^*$, and that otherwise a fail message is returned in line 7. Note that when $\vec{s}^*$ exists yet $\vec{s}^* < [0, 0, \ldots]$, the procedure returns a fail message in line 6.

Under the assumption $\vec{s}^*$ exists, the value of m computed line 1 is $m = \|\vec{s}\| - n = \|\vec{s}\| - \|\vec{s}^*\|$. From Lemma 9, there exists an index $0 \leq d^* \leq d_m$ such that $\vec{s}_{d^*-1} = \vec{s}^*_{d^*-1}$, and $s_{d^*} \neq s^*_{d^*}$. Since $\vec{s}^* \leq \vec{s}$, it must hold that $s^*_{d^*} < s_{d^*}$, and so $s^*_{d^*} \leq s_{d^*} - 1$. Therefore, $n = \|\vec{s}^*\| \geq \|\vec{s}^*_{d^*}\| = \|\vec{s}^*_{d^*-1}\| + 2^{d^*}\text{abs}(s^*_{d^*}) \geq \|\vec{s}_{d^*-1}\| + 2^{d^*}$ $\max\{-s_{d^*} + 1, 0\}$. In particular, d^* satisfies the condition in line 2. Consequentially, if the condition in line 2 does not hold, it contradicts the existence of $\vec{s}^*$, and the procedure indeed returns a fail message in this case (line 7).

Next, assume the condition in line 2 is met, and let $0 \leq d \leq d_m$ be the maximum index satisfying $n \geq \|\vec{s}_{d-1}\| + 2^d \max\{-s_d + 1, 0\}$, as selected in line 3. Thus, $d \geq d^*$. Denote $\delta = 1$ if $d = d_m$, and $\delta = 2$ if $d < d_m$. We will consider separately the two cases of computing the signature $\vec{s}'$ in lines 3–5. In both cases, the prefix $\vec{s}'_{d-1}$ of $\vec{s}'$ is set to be identical to the prefix $\vec{s}_{d-1}$ of $\vec{s}$.

Case 1: $\vec{s}'$ is computed according to lines 3 and 4. In this case, s'_d is set to $s_d - \delta$ in line 3, and s'_{d+1} is set to $\frac{n - \|\vec{s}'_d\|}{2^{d+1}}$in line 4. All values s'_i for $i > d + 1$ are implicitly set to zeros. Also, the condition $n \geq \|\vec{s}'_d\|$ holds in line 4, in particular $s'_{d+1} = \frac{n - \|\vec{s}'_d\|}{2^{d+1}} \geq 0$, and so abs $(s'_{d+1}) = s'_{d+1}$. Observe that $\vec{s}' \leq \vec{s}$ (since $\vec{s}'_{d-1} = \vec{s}_{d-1}$ and $s'_d < s_d$), and that $\|\vec{s}'\| = \|\vec{s}'_d\| + 2^{d+1}$ abs$(s'_{d+1} = \|\vec{s}'_d\| + n - \|\vec{s}'_d\| = n$.

By definition, $\vec{s}' \leq \vec{s}^* \leq \vec{s}$. Since $\vec{s}'_{d-1} = \vec{s}_{d-1}$, it follows that $\vec{s}'_{d-1} = \vec{s}^*_{d-1}$. In addition, since $d^* \leq d$ and since $s^*_{d^*} < s_{d^*}$, it must be that $d = d^*$. From Lemma 9, $s^*_d \leq s_d - \delta$, and since $s_d - \delta = s'_d \leq s^*_d \leq s_d - \delta$ it follows that $s'_d = s^*_d$, thus $\vec{s}'_d = \vec{s}^*_d$. Finally, since $s^*_{d+1} \leq$ abs$(s^*_{d+1}) = \frac{\|\vec{s}^*_{d+1}\| - \|\vec{s}^*_d\|}{2^{d+1}} \leq \frac{n - \|\vec{s}'_d\|}{2^{d+1}} = s'_{d+1}$, it follows that $s'_{d+1} = s^*_{d+1}$. Now, as $\vec{s}'_{d+1} = \vec{s}^*_{d+1}$, and since $\|\vec{s}'_{d+1}\| = \|\vec{s}^*_{d+1}\| = n$, it follows that for every $i > d + 1$, we have that $s'_i = s^*_i = 0$, and so $\vec{s}' = \vec{s}^*$.

Case 2: $\vec{s}'$ is computed according to lines 3 and 5. Here, the condition in line 4 does not hold, that is. (and due to the initialization of $\vec{s}'$ in lines 3–4), $n < \|\vec{s}_{d-1}\| + 2^d$ abs$(s_d - \delta)$. Now, s'_d is set to be $\frac{n - \|\vec{s}'_{d-1}\|}{2^d}$ in line 5, and all values s'_i for $i > d$ are implicitly set to zeros. We start by showing it that $s_d > 0$ in this case.

Assume by contradiction that $s_d \leq 0$. As $d \leq d_m$, the number $\frac{m}{2^d} = \frac{\|\vec{s}\| - n}{2^d}$ is an integer. We get that $\frac{\|\vec{s}_{d-1}\| + \sum_{i=d}^{\infty} 2^i \text{abs}(s_i) - n}{2^d} = \frac{\|\vec{s}_{d-1}\| - n}{2^d} + x$ is an integer for the integer $x = \frac{\sum_{i=d}^{\infty} 2^i \text{abs}(s_i)}{2^d}$, and so $\frac{n - \|\vec{s}_{d-1}\|}{2^d}$ is an integer. From the conditions in lines 2 and 4 and since $s_d \leq 0$ by assumption, $-s_d + 1 = \max\{-s_d + 1, 0\} \leq \frac{n - \|\vec{s}_{d-1}\|}{2^d} < -s_d + \delta$. Therefore, it must be that $\delta = 2$, and that $n = \|\vec{s}_{d-1}\| + 2^d(-s_d + 1) = \|\vec{s}_d\| + 2^d$. Moreover, since $\delta = 2$, it follows that $d < d_m$. Nevertheless, we get that $m = \|\vec{s}\| - n = \|\vec{s}_d\| + \sum_{i=d+1}^{\infty} 2^i \text{abs}(s_i) - \|\vec{s}_d\| - 2^d = \sum_{i=d+1}^{\infty} 2^i \text{abs}(s_i) - 2^d$. This implies that m does not divide by $2^{d+1} \leq 2^{d_m}$, in contradiction to the definition of d_m.

As we have established that $s_d > 0$, we can observe that $s'_d = \frac{n - \|\vec{s}_{d-1}\|}{2^d} < \frac{\|\vec{s}_{d-1}\| + 2^d \text{abs}(s_d - \delta) - \|\vec{s}_{d-1}\|}{2^d} = \text{abs}(s_d - \delta) \leq s_d$. Therefore, $\vec{s}' < \vec{s}$. It can be shown similarly as in Case 1 that $\|\vec{s}'\| = n$ and that $\vec{s}^* \leq \vec{s}'$, completing the proof. ■

Algorithm 5: MIN-INCREMENT ($\vec{s}, n$)

```
    Input: A signature $\vec{s} \leq [1, 0, \ldots]$ and an integer $n \geq 0$.
    Output: The lexicographically minimal signature $\vec{s} \leq \vec{s}' \leq [1, 0, \ldots]$ such that $\|\vec{s}'\| = n$, or the message
      "FAILED" if there is no such signature.
1   Let $m = \|\vec{s}\| - n$. If $m = 0$ then return $\vec{s}$.
2
3   Else if there is an integer $0 \leq d \leq d_m$ such that $n \geq \|\vec{s}_{d-1}\| + 2^d \max\{s_d + 1, 0\}$ then
4     Let $d$ be the maximum integer meeting the condition above. Initialize $\vec{s}'$ so that $\vec{s}'_{d-1} = \vec{s}_{d-1}$, and
      $s'_d = s_d + 2$ if $d < d_m$, or $s'_d = s_d + 1$ if $d = d_m$.
5     If $n \geq \|\vec{s}'_d\|$ then set $s'_{d+1} \leftarrow \frac{\|\vec{s}'_d\| - n}{2^{d+1}}$.
6
7     Else set $s'_d \leftarrow \frac{\|\vec{s}'_{d-1}\| - n}{2^d}$.
8
9     If $\vec{s}' \leq [1, 0, \ldots]$ then return $\vec{s}'$,
10    else return "FAILED."
11  Else return "FAILED."
```

The MIN-INCREMENT procedure is proven symmetrically, and its pseudocode is given in Algorithm 5. It is in fact a simplified version of the SIGNATURE-FOLD procedure in Zakov et al. (2013) (Supporting Information).

ACKNOWLEDGMENTS

The authors are thankful to the anonymous JCB and RECOMB reviewers for their helpful comments. The research was supported by grants from the NIH (RO1-HG004962) and the NSF (CCF-1115206, IIS-1318386).

AUTHOR DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Alkan, C., Kidd, J.M., Marques-Bonet, T., et al. 2009. Personalized copy number and segmental duplication maps using next-generation sequencing. *Nat. Genet.* 41, 1061–1067.

Bignell, G.R., Greenman, C.D., Davies, H., et al. 2010. Signatures of mutation and selection in the cancer genome. *Nature* 463, 893–898.

Bignell, G.R., Santarius, T., Pole, J.C., et al. 2007. Architectures of somatic genomic rearrangement in human cancer amplicons at sequence-level resolution. *Genome Res.* 17, 1296–1303.

Campbell, P.J., Yachida, S., Mudie, L.J., et al. 2010. The patterns and dynamics of genomic instability in metastatic pancreatic cancer. *Nature* 467, 1109–1113.

Carr, A.M., Paek, A.L., and Weinert, T. 2011. DNA replication: failures and inverted fusions. *Semin. Cell Dev. Biol.* 22, 866–874.

Chiang, D.Y., Getz, G., Jaffe, D.B., et al. 2009. High-resolution mapping of copy-number alterations with massively parallel sequencing. *Nat. Methods* 6, 99–103.

Eckel-Passow, J.E., Atkinson, E.J., Maharjan, S., et al. 2011. Software comparison for evaluating genomic copy number variation for Affymetrix 6.0 SNP array platform. *BMC Bioinform.* 12, 220.

Greenman, C., Bignell, G., Butler, A., et al. 2010. PICNIC: an algorithm to predict absolute allelic copy number variation with microarray cancer data. *Biostatistics* 11, 164–175.

Greenman, C., Cooke, S., Marshall, J., et al. 2012. Modelling breakage-fusion-bridge cycles as a stochastic paper folding process. *arXiv*. Available at: http://arxiv.org/abs/1211.2356

Hanahan, D., and Weinberg, R.A. 2011. Hallmarks of cancer: the next generation. *Cell* 144, 646–674.

Hastings, P.J., Lupski, J.R., Rosenberg, S.M., et al. 2009. Mechanisms of change in gene copy number. *Nat. Rev. Genet.* 10, 551–564.

Kinsella, M., and Bafna, V. 2012. Combinatorics of the breakage-fusionbridge mechanism. *J. Comput. Biol.* 19, 662–678.

Kitada, K., and Yamasaki, T. 2008. The complicated copy number alterations in chromosome 7 of a lung cancer cell line is explained by a model based on repeated breakage-fusion-bridge cycles. *Cancer Genet. Cytogenet.* 185, 11–19.

Knuth, D.E. 1998. *The Art of Computer Programming*, *Volume 3: Sorting and Searching*. International Monetary Fund, Washington, DC.

McClintock, B. 1938. The production of homozygous deficient tissues with mutant characteristics by means of the aberrant mitotic behavior of ring-shaped chromosomes. *Genetics* 23, 315–376.

McClintock, B. 1941. The stability of broken ends of chromosomes in zea mays. *Genetics* 26, 234–282.

Medvedev, P., Stanciu, M., and Brudno, M. 2009. Computational methods for discovering structural variation with next-generation sequencing. *Nat. Methods* 6, 13–20.

Olshen, A.B., Venkatraman, E., Lucito, R., et al. 2004. Circular binary segmentation for the analysis of array-based dna copy number data. *Biostatistics* 5, 557–572.

Pearl, J. 1984. *Heuristics*. Addison-Wesley Publishing Company, Reading, MA.

Reshmi, S., Roychoudhury, S., Yu, Z., et al. 2007. Inverted duplication pattern in anaphase bridges confirms the breakage-fusion-bridge (bfb) cycle model for 11q13 amplification. *Cytogenet. Genome Res.* 116, 46–52.

Sloane, N.J. 2007. The on-line encyclopedia of integer sequences, 130. *In Towards Mechanized Mathematical Assistants*. Springer, New York. Available at: http://oeis.org/A033485

Venkatraman, E.S., and Olshen, A.B. 2007. A faster circular binary seg-71 mentation algorithm for the analysis of array cgh data. *Bioinformatics* 23, 657–663.

Yoon, S., Xuan, Z., Makarov, V., et al. 2009. Sensitive and accurate detection of copy number variants using read depth of coverage. *Genome Res.* 19, 1586–1592.

Zakov, S., Kinsella, M., and Bafna, V. 2013. An algorithmic approach for breakage-fusion-bridge detection in tumor genomes. *Proc. Natl. Acad. Sci. USA* 110, 5546–5551.

Address correspondence to:
Dr. Shay Zakov
Department of Computer Science and Engineering
University of California–San Diego
9500 Gilman Drive
La Jolla, CA 92093

E-mail: szakov@eng.ucsd.edu