Supplementary Materials

6 Shouji Filter

6.1 Examining the Effect of Different Window Sizes on the Accuracy of the Shouji Algorithm.

In Fig. 4, we experimentally evaluate the effect of different window sizes on the false accept rate of Shouji. We observe that as we increase the window size, the rate of dissimilar sequences that are accepted by Shouji decreases. This is because individual matches (i.e., single zeros) are usually useless and they are not necessarily part of the common subsequences. As we increase the search window size, we are ignoring these individual matches and instead we only look for longer streaks of consecutive zeros. We also observe that a window size of 4 columns provides the lowest false accept rate (i.e., the highest accuracy).



Fig. 4: The effect of the window size on the rate of the falsely-accepted sequences (i.e., dissimilar sequences that are considered as similar ones by Shouji filter). We observe that a window width of 4 columns provides the highest accuracy. We also observe that as window size increases beyond 4 columns, more similar sequences are rejected by Shouji, which should be avoided.

6.2 The Shouji Algorithm and Its Analysis

We provide the Shouji algorithm along with analysis of its computational complexity (asymptotic run time and space complexity). Shouji divides the problem of finding the common subsequences into at most *m* subproblems, as described in Algorithm 1 (line 9). Each subproblem examines each of the 2E+1 bit-vectors and finds the 4-bit subsequence that has the largest number of zeros within the sliding window (line 13 to line 23). Once found, Shouji also compares the found subsequence with its corresponding subsequence in the Shouji bit-vector and stores the subsequence that has more zeros in the Shouji bit-vector (line 24). Now, let *c* be a constant representing the run time of examining a subsequence of 4 bits long. Then, the time complexity of the Shouji algorithm is as follows:

$$\Gamma_{Shouji}(m) = c.m.(2E+2) \tag{2}$$

This demonstrates that the Shouji algorithm runs in linear time with respect to the sequence length and edit distance threshold. The Shouji algorithm maintains 2E+1 diagonal bit-vectors and an additional auxiliary bit-vector (i.e., the Shouji bit-vector) for each two given sequences. The space complexity of the Shouji algorithm is as follows:

$$D_{Shouji}(m) = m . (2E+2) \tag{3}$$

Hence, the Shouji algorithm requires linear space with respect to the sequence length and edit distance threshold. Next, we describe the hardware implementation details of the Shouji filter.

6.3 Hardware Implementation

We present the FPGA chip layout for our hardware accelerator in Fig. 5. As we illustrated in the main manuscript, Section 2.3, we implement the first step of our Shouji algorithm, building neighborhood map, using shift registers and bitwise XOR operations. The second step of the Shouji algorithm is identifying the diagonally-consecutive matches. This key step involves finding the 4-bit vector that has the largest number of zeros. For each search window, there are 2E+1 diagonal bit-vectors and an additional Shouji bit-vector. To enable the computation to be performed in a parallel fashion, we build 2E+2 counters. As presented in Fig. 5, each counter counts the number of zeros in a single bit-vector. The counter takes four bits as input and generates three bits that represent the number of zeros within the window. Each counter requires three 4-input LUTs, as each LUT has a single output signal. In total, we need 6E+6 4-input LUTs to build a single search window. All bits of the counter output are generated at the same time, as the propagation delay through an FPGA look-up table is independent of the implemented function (Xilinx, November 17, 2014). The comparator is responsible for selecting the 4-bit subsequence that maximizes the number of consecutive matches based on the output of each counter and the Shouji bit-vector. Finally, the selected 4-bit subsequence is then stored in the Shouji bit-vector at the same corresponding location.

Input: text (7), pattern (P), edit distance threshold (E).Output: 1 (Similar/Alignment is needed) / 0 (Dissimilar/Alignment is not needed).1: $m \leftarrow length(T)$;Step 1: Building2: for $i \leftarrow 1$ to m doneighborhood map (N)3: for $j \leftarrow i-E$ to $i+E$ doOutput: 2E+1 diagonal5: $N[i,j] \leftarrow 0$;Dutput: 2E+1 diagonal6: else $N[i,j] \leftarrow 1$; //initializing Shouji bit-vector to 1'sbit-vectors8: $Z \leftarrow [0000]; // Z$ is 4-bit vector that stores the longest streak of diagonally-consecutive zeros9:9: for $i \leftarrow 1$ to m do // slide the search window by a single step10:10: for $j \leftarrow 1$ to E do // iterate over the diagonals11:11: // function CZ(D) counts the occurrence of zeros in its input bit-vector D12:12: // Compare j th lower diagonal with j th upper diagonal13:13: if CZ(N[i+j:H-3+j,i:H-3]) > CZ(N[i:H-3,i+j:H-3+j]) thenStep 2: Identifying the16: // zeros then selects the diagonal that starts with zerosDiagonally-Consecutive17: else if CZ(N[i+j:H-3+j,i:H-3]) = CZ(N[i:H-3,i+j:H-3+j]) thenMatches18: if N[i+j,j]==0 then Z ~ N[i+j:H-3+j];19:19: else if N[i,i+j]==0 then Z ~ N[i+j:H-3+j];10: // Compare Z with main diagonal and Shouji bit-vector11: else Z ~ N[i+j:A+3+j];12: // Compare Z with main diagonal and Shouji bit-vector13: if CZ(N[i+i+3,i+3+j])14: else Z < N[i+i+3,i+3+j];15: // Compare Z with main diagonal and Shouji bit-vector16: // Z(Z) > CZ(Shouji [i-i+3,i+3+j];17: else if CZ(N[i+i]) > CZ(Z) then Z ~ N[i+i+3,i+3];18: if CZ(N[i+i+	Algorithm 1: Shouji	Comments
Output: 1 (Similar/Alignment is not needed).1: $m \leftarrow length(T);$ Step 1: Building neighborhood map (N)2: for $i \leftarrow 1$ to m doneighborhood map (N)3: for $j \leftarrow i.E$ to $i+E$ doOutput: $2E+1$ diagonal bit-vectors5: $N[i,j] \leftarrow 0;$ Dutput: $2E+1$ diagonal bit-vectors6: else $N[i,j] \leftarrow 1;$ i' if $T = 1$ to m do Shouji[$i] \leftarrow 1;$ 7: for $i \leftarrow 1$ to m do Shouji[$i] \leftarrow 1;$ i' intializing Shouji bit-vector to 1's8: $Z \leftarrow [0000]; // Z$ is 4-bit vector that stores the longest streak of diagonally-consecutive zeros9: for $i \leftarrow 1$ to m do // slide the search window by a single step10: for $j \leftarrow 1$ to E do // iterate over the diagonals11: // function $CZ(D)$ counts the occurrence of zeros in its input bit-vector D12: // Compare j^{h} lower diagonal with j^{h} upper diagonal13: if $CZ(N[i+j:i+3+j,i:+3]) > CZ(N[i:i+3,i+j:i+3+j])$ then14: $Z \leftarrow N[i+j:i+3+j,i:+3]$;15: // if j^{h} lower and j^{h} upper diagonal that starts with zeros17: else if $CZ(N[i+j:i+3+j,i:i+3]) = CZ(N[i:i+3,i+j:i+3+j])$ then18: if $N[i+j,j] == 0$ then $Z \leftarrow N[i+j:i+3+j]$;19: else if $N[i,j+j] == 0$ then $Z \leftarrow N[i+i+3+j,i:i+3+j]$;19: else if $N[i,j+j] == 0$ then $Z \leftarrow N[i+i+3+j,i:i+3+j]$;10: // Compare Z with main diagonal and Shouji bit-vector11: else $Z \leftarrow N[i+i+3,i+3+j]$;12: // Compare Z with main diagonal and Shouji bit-vector13: else $Z < N[i+i+3,i+3+j]$;14: else $Z \leftarrow N[i+i+3,i+3+j]$;15: // Compare Z with main diagonal and Shouji bit-vector16: // CZ(D) count $Z \leftarrow N[i+i+3,i+3+j]$;17: else if $CZ(N[i+i$	Input: text (T), pattern (P), edit distance threshold (E).	
1: $m \leftarrow length(T)$;Step 1: Building neighborhood map (N)3:for $i \leftarrow l \neq to$ mdo3:for $j \leftarrow i \neq t$ to mdo 3:for $j \leftarrow i \neq t$ to mdo 4:if $T(I) = P[I]$ then5: $N(i,j) \leftarrow 0$;6:else $N(i,j) \leftarrow 1$;7:for $i \leftarrow 1$ to $m do$ Shouji[$I \leftarrow 1$; //initializing Shouji bit-vector to 1's8: $Z \leftarrow [0000]; //Z is 4-bit vector that stores the longest streak of diagonally-consecutive zeros9:for i \leftarrow 1 to m do //slide the search window by a single step10:for j \leftarrow 1 to E do // iterate over the diagonals11:// function C2(D) counts the occurrence of zeros in its input bit-vector D12:// Compare j^{th} lower diagonal with j^{th} upper diagonal13:if C2(N[i+i;1+3+j,i:+3])14:Z \leftarrow N[i+j;1+3+j,i:+3];15:// If j^{th} lower and j^{th} upper diagonals have the same number of16:// zeros then selects the diagonal that starts with zeros18:if N[i+j,1]==0 then Z \leftarrow N[i+i;3,i+j:i+3+j] then18:if N[i+j,1]==0 then Z \leftarrow N[i+i;3,i+j:i+3+j];20:// Compare Z with the j^{th} upper diagonal21:else if N[i+j]==0 then Z \leftarrow N[i+i;3,i+j:i+3+j];20:// Compare Z with main diagonal and Shouji bit-vector23:if C2(N[i:i+3,i+3]) > CZ(2 then Z \leftarrow N[i:i+3,i+j];24:if C2(Z) > CZ(Shouji[i:i+3]) then Shouji[i:i+3] \leftarrow Z;25: th C2(Shouji]) \geq m-E then return 1;Step 3: Filtering out$	Output : 1 (Similar/Alignment is needed) / 0 (Dissimilar/Alignment is not needed).	
2: for $i \leftarrow 1$ to m doneighborhood map (N)3:for $j \leftarrow i$ -E to i +E doneighborhood map (N)4:if $T[i] = P[j]$ thenOutput: 2E+1 diagonal5: $N[i,j] \leftarrow 0$;output: 2E+1 diagonal6:else $N[i,j] \leftarrow 1$;bit-vectors7: for $i \leftarrow 1$ to m do Shouji[$i] \leftarrow 1$; //initializing Shouji bit-vector to 1'sbit-vectors8: $Z \leftarrow [0000]; // Z$ is 4-bit vector that stores the longest streak of diagonally-consecutive zerosbit-vectors9: for $i \leftarrow 1$ to m do // slide the search window by a single stepfor $i \leftarrow 1$ to m do // iterate over the diagonals11:// function C2(D) counts the occurrence of zeros in its input bit-vector D12:// Compare \mathbb{I}^n lower diagonal with \mathbb{I}^n upper diagonal13:if $CZ(N[i+j:i+3+j],i:i+3]$) $CZ(N[i:i+3,i+j:i+3+j])$ then14: $Z \leftarrow N[i+j:i+3+j,i:i+3]$ 15:// If \mathbb{I}^n lower and \mathbb{I}^n upper diagonal that starts with zeros16:// zeros then selects the diagonal that starts with zeros17:else if $CZ(N[i+j:i+3+j,i:i+3]) = CZ(N[i:i+3,i+j:i+3+j])$ then18:if $N[i,i+j] = 0$ then $Z \leftarrow N[i:i+3,i+j:i+3+j]$ 19:else if $N[i,i+j] = 0$ then $Z \leftarrow N[i:i+3,i+j:i+3+j]$ 20:// Compare Z with main diagonal and Shouji bit-vector21:else $Z \leftarrow N[i:i+3,i+i+3+j]$;22:// Compare Z with main diagonal and Shouji bit-vector23:if $CZ(Z) > CZ(Shoujj[i:i+3]$ then $Shoujj[i:i+3] \leftarrow Z$;24:if $CZ(Z) < CZ(Z)$ then $Z \leftarrow N[i:i+3,i+3];$ 24:if $CZ(Z) < CZ(Z)$ then $Z \leftarrow N[i:i+3,i+3]; \leftarrow Z$;25: <td< td=""><td>1: $m \leftarrow length(T);$</td><td>Step 1: Building</td></td<>	1: $m \leftarrow length(T);$	Step 1: Building
3:for $j \leftarrow i$ - E to i + E doOutput: 2 E +1 diagonal4:if $T[i] == P[i]$ thenOutput: 2 E +1 diagonal5: $N[i,j] \leftarrow 0;$ bit-vectors6:else $N[i,i] \leftarrow 1;$ if it is in the intervent of the interv	2: for $i \leftarrow 1$ to m do	neighborhood map (N)
4:if $T[i] == P[i]$ thenOutput: $2E+1$ diagonal5: $N[i,j] \leftarrow 0;$ bit -vectors6:else $N[i,j] \leftarrow 1;$ bit -vector to 1's7:for $i \leftarrow 1$ to m do Shouj $[i] \leftarrow 1;$ //initializing Shouji bit-vector to 1's8: $Z \leftarrow [0000]; // Z$ is 4-bit vector that stores the longest streak of diagonally-consecutive zeros9:for $i \leftarrow 1$ to m do // slide the search window by a single step10:for $j \leftarrow 1$ to E do // iterate over the diagonals11:// function CZ(D) counts the occurrence of zeros in its input bit-vector D12:// Compare J^{ih} lower diagonal with J^{ih} upper diagonal13:if $CZ(N[i+j:i+3+j,i:+3]) > CZ(N[i:i+3,i+j:i+3+j])$ then14: $Z \leftarrow N[i+j:i+3+j]$ upper diagonals have the same number of15:// If J^{ih} lower and J^{ih} upper diagonal that starts with zeros16:// zeros then selects the diagonal that starts with zeros17:else if $CZ(N[i+j:i+3+j,i:i+3]) == CZ(N[i:i+3,i+j:i+3+j])$ then18:if $N[i+j,i] == 0$ then $Z \leftarrow N[i+j:i+3+j]$;19:else if $N[i,i+j] == 0$ then $Z \leftarrow N[i:i+3,i+j:i+3+j]$;20:// Compare Z with the J^{ih} upper diagonal21:else $Z \leftarrow N[i:i+3,i+i+3+j]$;22:// Compare Z with main diagonal and Shouji bit-vector23:if $CZ(N[i:i+3,i+3]) > CZ(2)$ then $Z \leftarrow N[i:i+3,i:i+3];$ 24:if $CZ(Z) > CZ(shouj[i:i+3])$ then Shouj[i[i:i+3] $\leftarrow Z;$ 25: if $CZ(Shoujj) \ge m-E$ then return 1;25: if $CZ(Shoujj) \ge m-E$ then return 1;	3: for $j \leftarrow i-E$ to $i+E$ do	
5: $N[i,j] \leftarrow 0;$ bit-vectors6:else $N[i,j] \leftarrow 1;$ bit-vectors7:for $i \leftarrow 1$ to m do Shouji $[i] \leftarrow 1;$ //initializing Shouji bit-vector to 1'ss. $Z \leftarrow [0000]; // Z$ is 4-bit vector that stores the longest streak of diagonally-consecutive zeros9:for $i \leftarrow 1$ to m do // slide the search window by a single step10:10:for $j \leftarrow 1$ to E do // iterate over the diagonals11:11:// function CZ(D) counts the occurrence of zeros in its input bit-vector D12:// Compare j^{th} lower diagonal with j^{th} upper diagonal13:if $CZ(N[i+j:i+3+j,i:i+3])$ 14: $Z \leftarrow N[i+j:i+3+j,i:i+3];$ 15:// If j^{th} lower and j^{th} upper diagonals have the same number of16:// zeros then selects the diagonal that starts with zeros17:else if $CZ(N[i+j:i+3+j,i:i+3]) = CZ(N[i:i+3,i+j:i+3+j])$ then18:if $N[i+j,i]==0$ then $Z \leftarrow N[i+j:i+3+j,i:i+3];$ 19:else if $N[i,i+j]==0$ then $Z \leftarrow N[i+j:i+3+j];$ 20:// Compare Z with the j^{th} upper diagonal21:else $Z \leftarrow N[i+j:i+3+j];$ 22:// Compare Z with main diagonal and Shouji bit-vector23:if $CZ(N[i:i+3,i+3+j]) > CZ(2)$ then $Z \leftarrow N[i:i+3,i+i+3+j];$ 24:if $CZ(Z) > CZ(Shouji](i:+3]$ then Shouji[(i:+3] $\leftarrow Z;$ 25:if $CZ(N_{i}) > ZZ(Z)$ then $Z \leftarrow N[i:i+3,i:i+3];$ 24:if $CZ(Z) > CZ(Shoujji[i:+3]$ then Shoujj[(i:+3] $\leftarrow Z;$ 25:if $CZ(Shoujji > m-E$ then return 1;	4: if $T[i] == P[j]$ then	Output: 2E+1 diagonal
6:else N(i,j) ← 1;7:for i ← 1 to m do Shouji[i] ← 1; //initializing Shouji bit-vector to 1's8: Z ← [0000]; // Z is 4-bit vector that stores the longest streak of diagonally-consecutive zeros9:for i ← 1 to m do // slide the search window by a single step10:for j ← 1 to E do // iterate over the diagonals11:// function CZ(D) counts the occurrence of zeros in its input bit-vector D12:// Compare j th lower diagonal with j th upper diagonal13:if CZ(N[i+j:i+3+j,i:i+3]) > CZ(N[i:i+3,i+j:i+3+j]) then14:Z ← N[i+j:i+3+j,i:i+3];15:// If j th lower and j th upper diagonal that starts with zeros16:// zeros then selects the diagonal that starts with zeros17:else if CZ[N[i+j:i+3+j,i:i+3]] == CZ(N[i:i+3,i+j:i+3+j]) then18:if N[i+j,i]==0 then Z ← N[i:i+3,i+j:i+3+j];19:else if N[i,i+j]==0 then Z ← N[i:i+3,i+j:i+3+j];20:// Compare Z with the j th upper diagonal21:else Z ← N[i:i+3,i+j:i+3+j];22:// Compare Z with main diagonal and Shouji bit-vector23:if CZ[N[i:i+3,i+i+3]) > CZ[2] then Z ← N[i:i+3,i+i+3];24:if CZ[Z) > CZ[Shouji[i:i+3]) then Shouji[i:i+3] ← Z;25:if CZ[Shouji[i:i+3]) then shouji[i:i+3] ← Z;25:if CZ[Shouji] ≥ m-E then return 1;	5: $N[i,j] \leftarrow 0;$	bit-vectors
7: for $l \leftarrow 1$ to m do Snoul/[1] $\leftarrow 1$; //initializing Snoul/ bit-Vector to 1.38: $Z \leftarrow [0000]$; // Z is 4-bit vector that stores the longest streak of diagonally-consecutive zeros9: for $i \leftarrow 1$ to m do // slide the search window by a single step10: for $j \leftarrow 1$ to E do // iterate over the diagonals11: // function CZ(D) counts the occurrence of zeros in its input bit-vector D12: // Compare j th lower diagonal with j th upper diagonal13: if CZ(N[i+j:i+3+j,i:i+3]) > CZ(N[i:i+3,i+j:i+3+j]) then14: $Z \leftarrow N[i+j:i+3+j,i:i+3]$;15: // If j th lower and j th upper diagonals have the same number of16: // zeros then selects the diagonal that starts with zeros17: else if $CZ(N[i+j:i+3+j,i:i+3]) = CZ(N[i:i+3,i+j:i+3+j])$ then18: if $N[i+j,i] = 0$ then $Z \leftarrow N[i+j:i+3+j,i:i+3]$;19: else if $N[i,i+j] = 0$ then $Z \leftarrow N[i:i+3,i+j:i+3+j]$;20: // Compare Z with the j th upper diagonal21: else Z $\leftarrow N[i:i+3,i+j:i+3+j]$;22: // Compare Z with main diagonal and Shouji bit-vector23: if $CZ(N[i:i+3,i:i+3]) > CZ(Z]$ then $Z \leftarrow N[i:i+3,i+j:i+3]$;24: if $CZ(Z) > CZ(Shouji[i:i+3])$ then Shouji[i:i+3] $\leftarrow Z$;25: if CZ(Shouji) $\ge m-E$ then return 1;	b: else $N[i,j] \leftarrow 1$;	
8: $2 \leftarrow [0000]; // 2$ is 4-bit vector that stores the longest streak of aliagonally-consecutive zeros 9: for $i \leftarrow 1$ to m do // slide the search window by a single step 10: for $j \leftarrow 1$ to E do // iterate over the diagonals 11: // function CZ(D) counts the occurrence of zeros in its input bit-vector D 12: // Compare j^{th} lower diagonal with j^{th} upper diagonal 13: if CZ(N[i+j:i+3+j,i:i+3]) > CZ(N[i:i+3,i+j:i+3+j]) then 14: $Z \leftarrow N[i+j:i+3+j,i:i+3]$; 15: // If j^{th} lower and j^{th} upper diagonals have the same number of 16: // zeros then selects the diagonal that starts with zeros 17: else if $CZ(N[i:i+3+j,i:i+3]) == CZ(N[i:i+3,i+j:i+3+j])$ then 18: if $N[i+j,i]==0$ then $Z \leftarrow N[i+j:i+3+j,i:i+3+j]$; 19: else if $N[i,i+j]==0$ then $Z \leftarrow N[i+j:i+3+j];$ 20: // Compare Z with the j^{th} upper diagonal 21: else $Z \leftarrow N[i:i+3,i+j:i+3+j];$ 22: // Compare Z with main diagonal and Shouji bit-vector 23: if $CZ(N[i:i+3,i:i+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:i+3];$ 24: if $CZ(Z) > CZ(Shouji[i:i+3])$ then Shouji[i:i+3] $\leftarrow Z;$ 25: if $CZ(Shouji) \ge m-E$ then return 1; 5: Step 3: Filtering out	7: for $i \leftarrow 1$ to m do Shouji[i] $\leftarrow 1$; //initializing Shouji bit-vector to 1's	
9: for $i \leftarrow 1$ to m do // since the search window by a single step10:for $j \leftarrow 1$ to E do // iterate over the diagonals11:// function CZ(D) counts the occurrence of zeros in its input bit-vector D12:// Compare j th lower diagonal with j th upper diagonal13:if CZ(N[i+j:i+3+j,i:i+3]) > CZ(N[i:i+3,i+j:i+3+j]) then14: $Z \leftarrow N[i+j:i+3+j,i:i+3];$ 15:// If j th lower and j th upper diagonals have the same number of16:// zeros then selects the diagonal that starts with zeros17:else if CZ(N[i+j:i+3+j,i:i+3]) == CZ(N[i:i+3,i+j:i+3+j]) then18:if $N[i+j,i]==0$ then $Z \leftarrow N[i+j:i+3+j];$ 19:else if $N[i+j,i]==0$ then $Z \leftarrow N[i:i+3,i+j:i+3+j];$ 20:// Compare Z with the j th upper diagonal21:else $Z \leftarrow N[i:i+3,i+j:j;];$ 22:// Compare Z with main diagonal and Shouji bit-vector23:if $CZ(N[i:i+3,i:+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:+3];$ 24:if $CZ(Shouji[:i+3])$ then Shouji[i:i+3] $\leftarrow Z;$ 25: if $CZ(Shouji] \ge m-E$ then return 1;Step 3: Filtering out	8: $2 \leftarrow [0000]$; // 2 is 4-bit vector that stores the longest streak of alagonally-consecutive zeros	
10:10: $f \in T$ to F do f when the adagonals11:// function CZ(D) counts the occurrence of zeros in its input bit-vector D12:// Compare j^{th} lower diagonal with j^{th} upper diagonal13:if $CZ(N[i+j:i+3+j,i:i+3]) > CZ(N[i:i+3,i+j:i+3+j])$ then14: $Z \leftarrow N[i+j:i+3+j,i:i+3];$ 15:// If j^{th} lower and j^{th} upper diagonals have the same number of16:// zeros then selects the diagonal that starts with zeros17:else if $CZ(N[i+j:i+3+j,i:i+3]) = CZ(N[i:i+3,i+j:i+3+j])$ then18:if $N[i+j,i]==0$ then $Z \leftarrow N[i+j:i+3+j,i];$ 19:else if $N[i+j]==0$ then $Z \leftarrow N[i:i+3,i+j:i+3+j];$ 20:// Compare Z with the j^{th} upper diagonal21:else $Z \leftarrow N[i:i+3,i+j:i+3+j];$ 20:// Compare Z with main diagonal and Shouji bit-vector23:if $CZ(N[i:i+3,i:+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:+3];$ 24:if $CZ(Shouji[:i+3])$ then Shouji[i:i+3] $\leftarrow Z;$ 25:if $CZ(Shouji] \ge m-E$ then return 1;	9: for $i \leftarrow 1$ to m do // slide the search window by a single step	
11.7/ Junction (2/b) counts the obcurrence of zeros in its input bit-vector b12.// Compare j th lower diagonal with j th upper diagonal13.if $CZ(N[i+j:i+3+j,i:i+3]) > CZ(N[i:i+3,i+j:i+3+j])$ then14. $Z \leftarrow N[i+j:i+3+j,i:i+3];$ 15.// If j th lower and j th upper diagonals have the same number of16.// zeros then selects the diagonal that starts with zeros17.else if $CZ(N[i+j:i+3+j,i:i+3]) = CZ(N[i:i+3,i+j:i+3+j])$ then18.if $N[i+j,i]==0$ then $Z \leftarrow N[i+j:i+3+j,i];$ 19.else if $N[i+j,i]==0$ then $Z \leftarrow N[i:i+3,i+j:i+3+j];$ 20.// Compare Z with the j th upper diagonal21.else $Z \leftarrow N[i:i+3,i+j:i+3+j];$ 22.// Compare Z with main diagonal and Shouji bit-vector23.if $CZ(N[i:i+3,i:+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:+3];$ 24.if $CZ(Shouji[:i+3])$ then Shouji[i:i+3] $\leftarrow Z;$ 25.if $CZ(Shouji] \ge m-E$ then return 1;	10. Ior $f \leftarrow 1$ to E do f iterate over the accurrence of zeros in its input bit vector D	
12.if CZ(Ni[i+j:i+3+j,i:i+3]) > CZ(N[i:i+3,i+j:i+3+j]) then13.if CZ(N[i+j:i+3+j,i:i+3]) > CZ(N[i:i+3,i+j:i+3+j]) then14. $Z \leftarrow N[i+j:i+3+j,i:i+3];$ 15.// If j th lower and j th upper diagonals have the same number of16.// zeros then selects the diagonal that starts with zeros17.else if CZ(N[i+j:i+3+j,i:i+3]) == CZ(N[i:i+3,i+j:i+3+j]) then18.if N[i+j,i]==0 then Z \leftarrow N[i+j:i+3+j,i]; then19.else if N[i+j]==0 then Z \leftarrow N[i:i+3,i+j:i+3+j];20.// Compare Z with the j th upper diagonal21.else Z \leftarrow N[i:i+3,i+j:j];22.// Compare Z with main diagonal and Shouji bit-vector23.if CZ(N[i:i+3,i:+3]) > CZ(Z) then Z \leftarrow N[i:i+3,i:+3];24.if CZ(Z) > CZ(Shouji[:i+3]) then Shouji[i:i+3] ← Z;25.if CZ(Shouji] ≥ m-E then return 1;	11. // Junction C2(D) counts the occurrence of zeros in its input bit-vector D	
14: $Z \leftarrow N[i+j:i+3+j,i+3+j]$;Step 2: Identifying the15:// If j^{th} lower and j^{th} upper diagonals have the same number ofStep 2: Identifying the16:// zeros then selects the diagonal that starts with zerosDiagonally-Consecutive17:else if $CZ(N[i+j:+3+j,i:+3]) == CZ(N[i:i+3,i+j:i+3+j])$ thenMatches18:if $N[i+j,i]==0$ then $Z \leftarrow N[i+j:i+3+j,i:+3];$ Matches19:else if $N[i+j,i]==0$ then $Z \leftarrow N[i:i+3,i+j:i+3+j];$ Matches20:// Compare Z with the j^{th} upper diagonalMatches21:else $Z \leftarrow N[i:i+3,i+j:j];$ // Compare Z with main diagonal and Shouji bit-vector23:if $CZ(N[i:i+3,i:+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:+3];$ Step 3: Filtering out24:if $CZ(Shouji] \ge m-E$ then return 1;Step 3: Filtering out	12. <i>if</i> $C7/N[i+i:i+3+i:i:i+3] > C7/N[i:i+3:i+i:i+3+i]$ then	
15.// If j^{th} lower and j^{th} upper diagonals have the same number ofStep 2: Identifying the16.// zeros then selects the diagonal that starts with zerosDiagonally-Consecutive17.else if $CZ(N[i:j:i+3+j,i:i+3]) == CZ(N[i:i+3,i+j:i+3+j])$ thenMatches18.if $N[i+j,]==0$ then $Z \leftarrow N[i+j:i+3+j,i:i+3];$ Matches19.else if $N[i,i+j]==0$ then $Z \leftarrow N[i+j:i+3+j];$ Matches20.// Compare Z with the j^{th} upper diagonalMatches21.else $Z \leftarrow N[i:i+3,i+j:i+3+j];$ Matches22.// Compare Z with main diagonal and Shouji bit-vectorMatches23.if $CZ(N[i:i+3,i:i+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:i+3];$ Step 3: Filtering out24.if $CZ(Shouji] \ge m-E$ then return 1;Step 3: Filtering out	14: $7 \leftarrow N[i+i:i+3]$:	
16:// zeros then selects the diagonal that starts with zerosDiagonally-Consecutive17:else if $CZ(N[i+j:i+3+j,i:i+3]) == CZ(N[i:i+3,i+j:i+3+j])$ thenMatches18:if $N[i+j,i]==0$ then $Z \leftarrow N[i+j:i+3+j,i:i+3];$ Matches19:else if $N[i+j,i]==0$ then $Z \leftarrow N[i+j:i+3+j];$ Matches20:// Compare Z with the j th upper diagonalMatches21:else $Z \leftarrow N[i:i+3,i+j:i+3+j];$ Matches22:// Compare Z with main diagonal and Shouji bit-vectorMatches23:if $CZ(N[i:i+3,i:i+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:i+3];$ Matches24:if $CZ(S > CZ(Shouji[i:i+3])$ then Shouji[i:i+3] $\leftarrow Z;$ Step 3: Filtering out	15: // If i^{th} lower and i^{th} upper diagonals have the same number of	Step 2: Identifying the
17:else if $CZ(N[i+j:i+3+j,i:i+3]) == CZ(N[i:i+3,i+j:i+3+j])$ thenMatches18:if $N[i+j,i]==0$ then $Z \leftarrow N[i+j:i+3+j,i:i+3];$ Matches19:else if $N[i,i+j]==0$ then $Z \leftarrow N[i+j:i+3+j];$ Matches20:// Compare Z with the j th upper diagonalMatches21:else Z $\leftarrow N[i:i+3,i+j:i+3+j];$ Matches22:// Compare Z with main diagonal and Shouji bit-vectorMatches23:if $CZ(N[i:i+3,i:i+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:i+3];$ Matches24:if $CZ(S > CZ(Shouji[i:i+3])$ then Shouji[i:i+3] $\leftarrow Z;$ Step 3: Filtering out	16: // zeros then selects the diagonal that starts with zeros	Diagonally-Consecutive
18:if $N[i+j,i]==0$ then $Z \leftarrow N[i+j:i+3+j,j:i+3];$ 19:else if $N[i,i+j]==0$ then $Z \leftarrow N[i:i+3,i+j:i+3+j];$ 20:// Compare Z with the jth upper diagonal21:else $Z \leftarrow N[i:i+3,i+j:i+3+j];$ 22:// Compare Z with main diagonal and Shouji bit-vector23:if $CZ(N[i:i+3,i:i+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:i+3];$ 24:if $CZ(Shouji[:i:+3])$ then Shouji[i:i+3] $\leftarrow Z;$ 25:if $CZ(Shouji] \ge m-E$ then return 1;Step 3: Filtering out	17: else if $CZ(N[i+i:i+3+i,i:i+3]) == CZ(N[i:i+3,i+i:i+3+i])$ then	Matches
19:else if $N[i, j+j] == 0$ then $Z \leftarrow N[i: j+3, j+j: j+3+j]$;20:// Compare Z with the jth upper diagonal21:else $Z \leftarrow N[i: j+3, j+j: j+3+j]$;22:// Compare Z with main diagonal and Shouji bit-vector23:if $CZ(N[i: j+3, i: j+3]) > CZ(Z)$ then $Z \leftarrow N[i: j+3, i: j+3]$;24:if $CZ(Shouji[i: j+3])$ then Shouji[i: j+3] $\leftarrow Z$;25:if $CZ(Shouji] \ge m-E$ then return 1;Step 3: Filtering out	18: if $N[i+j,i] == 0$ then $Z \leftarrow N[i+j;i+3+j,j:i+3];$	
20:// Compare Z with the jth upper diagonal21:else Z $\leftarrow N[i:i+3,i+j:j+3+j];$ 22:// Compare Z with main diagonal and Shouji bit-vector23:if $CZ(N[i:i+3,i:i+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:i+3];$ 24:if $CZ(Z) > CZ(Shouji[i:i+3])$ then Shouji[i:i+3] $\leftarrow Z;$ 25:if $CZ(Shouji) \ge m-E$ then return 1;Step 3: Filtering out	19: else if $N[i,i+j] == 0$ then $Z \leftarrow N[i:i+3,i+j:i+3+j];$	
21:else $Z \leftarrow N[i:i+3,i+j:i+3+j];$ 22:// Compare Z with main diagonal and Shouji bit-vector23:if $CZ(N[i:i+3,i:i+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:i+3];$ 24:if $CZ(Z) > CZ(Shouji[i:i+3])$ then Shouji[i:i+3] $\leftarrow Z;$ 25:if $CZ(Shouji) \ge m-E$ then return 1;Step 3: Filtering out	20: // Compare Z with the j th upper diagonal	
22:// Compare Z with main diagonal and Shouji bit-vector23:if $CZ(N[i:i+3,i:i+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:i+3];$ 24:if $CZ(Z) > CZ(Shouji[i:i+3])$ then $Shouji[i:i+3] \leftarrow Z;$ 25:if $CZ(Shouji) \ge m-E$ then return 1;Step 3: Filtering out	21: else $Z \leftarrow N[i:i+3,i+j:i+3+j];$	
23:if $CZ(N[i:i+3,i:i+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:i+3];$ 24:if $CZ(Z) > CZ(Shouji[i:i+3])$ then $Shouji[i:i+3] \leftarrow Z;$ 25:if $CZ(Shouji) \ge m-E$ then return 1;Step 3: Filtering out	22: // Compare Z with main diagonal and Shouji bit-vector	
24:if $CZ(Z) > CZ(Shouji[::i+3])$ then $Shouji[::i+3] \leftarrow Z$;25:if $CZ(Shouji) \ge m-E$ then return 1;Step 3: Filtering out	23: if $CZ(N[i:i+3,i:i+3]) > CZ(Z)$ then $Z \leftarrow N[i:i+3,i:i+3];$	
25: if <i>Cz(Shouji)</i> ≥ <i>m</i> - <i>E</i> then return 1; Step 3: Filtering out	24: if $CZ(Z) > CZ(Shouji[i:i+3])$ then Shouji[i:i+3] $\leftarrow Z$;	
	25: if $CZ(Shouji) \ge m$ -E then return 1;	Step 3: Filtering out
26: else return 0; Dissimilar Sequences	26: else return 0;	Dissimilar Sequences

Algorithm 2: CZ (count zeros) functionFunction: CZ() counts the number of occurrences of zeros.Input: bit-vector D.Output: number of occurrences of zeros.1: $count \leftarrow 0;$ 2: for $i \leftarrow 1$ to length(D) do3: if D[i] == 0 then4: $count \leftarrow count + 1;$ 5: return count;

2



Fig. 5: FPGA chip layout for Shouji and block diagram of the search window scheme implemented in a Xilinx VC709 FPGA for a single filtering unit.

7 MAGNET Filter

First, we provide the MAGNET (Alser et al., July 2017) algorithm and describe its main filtering mechanism. Second, we analyze the computational complexity of the MAGNET algorithm. Third, we provide details about the hardware implementation of the MAGNET algorithm.

7.1 Overview

MAGNET (Alser et al., July 2017) is another filter that uses a divide-and-conquer technique to find all the E+1 common subsequences, if any, and sum up their length. By calculating their total length, we can estimate the total number of edits between the two given sequences. If the total length of the E+1common subsequences is less than *m*-*E*, then there exist more common subsequences than E+1 that are associated with more edits than allowed. If so, then MAGNET excludes the two given sequences from optimal alignment calculation. We present the algorithm of MAGNET in Algorithm 3.

Algorithm 3: MAGNET	Comments
Input: text (T), pattern (P), edit distance threshold (E).	
Output : 1 (Similar/Alignment is needed) / 0 (Dissimilar/Alignment is not needed).	
1: $m \leftarrow length(T);$ 2: for $i \leftarrow 1$ to m do 3: for $j \leftarrow i-E$ to $i+E$ do 4: if $T[i] == P[j]$ then	Step 1 : Building neighborhood map (<i>N</i>) Output: 2 <i>E</i> +1 diagonal
5: $N[i,j] \leftarrow 0;$ 6: else $N[i,j] \leftarrow 1;$	bit-vectors
7: for 1 ← 1 to m do 8: MAGNET[i] ← 1; // Initializing MAGNET bit-vector 9: [MAGNET, calls] ← EXEN(N, 1, m, E, MAGNET, 1);	Step 2 - Step 4
10: // Function CZ() returns number of zeros 11: if CZ(MAGNET) $\ge m$ -E then return 1; else return 0;	Step 5: Filtering out Dissimilar Sequences

Finding the common subsequences involves four main steps. (1) **Building the neighborhood map**. Similar to Shouji, MAGNET starts with building the 2E+1 diagonal bit-vectors of the neighborhood map for the two given sequences (Algorithm 3, lines 2-6). (2) **Extraction**. Each diagonal bit-vector nominates its local longest subsequence of consecutive zeros. Among all nominated subsequences, a single subsequence is selected as a global longest

subsequence based on its length (Algorithm 4, lines 2-11). MAGNET evaluates if the length of the global longest subsequence is less than [(m - E)/(E + 1)], then the two sequences contain more edits than allowed, which cause the common subsequences to be shorter (i.e., each edit results in dividing the sequence pair into more common subsequences). If so, then the two sequences are rejected (Algorithm 4, lines 12-13). Otherwise, MAGNET stores the length of the global longest subsequence to be used towards calculating the total length of all E+1 common subsequences. The lower bound equality occurs when all edits are equispaced and all E+1 subsequences are of the same length. (3) Encapsulation. The next step is essential to preserve the original edit (or edits) that causes a single common sequence to be divided into smaller subsequences. MAGNET penalizes the found subsequence by two edits (one for each side). This is achieved by excluding from the search space of all bit-vectors the indices of the found subsequence in addition to the index of the surrounding single bit from both left and right sides (Algorithm 4, lines 14-17). (4) Divide-and-Conquer Recursion. In order to locate the other E non-overlapping subsequences, MAGNET applies a divide-and-conquer technique where we decompose the problem of finding the non-overlapping common subsequences into two subproblems. While the first subproblem focuses on finding the next long subsequence that is located on the right-hand side of the previously found subsequence in the first extraction step (Algorithm 4, line 15), the second subproblem focuses on the other side of the found subsequence (Algorithm 4, line 17). Each subproblem is solved by recursively repeating all the three steps mentioned above, but without evaluating again the length of the longest subsequence. MAGNET applies two early termination methods that aim to reduce the execution time of the filter. The first method is evaluating the length of the longest subsequence in the first recursion call (Algorithm 4, lines 12-13). The second method is limiting the number of the subsequences to be found to at most E+I, regardless of their actual number for the given sequence pair (Algorithm 4, line 1). (5) Filtering out Dissimilar Sequences. Once after the termination, if the total length of all found common subsequences is less than m-E, then the two sequences are rejected. Otherwise, they are considered to be similar and the alignment can be measured using sophisticated alignment algorithms.

Algorithm 4: EXEN function	Comments
Function: EXEN() extracts the longest subsequence of consecutive zeros and generate two	
subproblems.	
Input: Neighborhood map (N), start index (SI), end index (EI), E, MAGNET bit-vector, number of	
recursion calls.	
Output: updated MAGNET bit-vector, updated number of calls.	
1: if (SI ≤ EI and calls ≤ E+1) then // Early termination condition	
2: // Function CCZ() returns number and indices of longest	
3: // subsequence of diagonally consecutive zeros	
4: for $j \leftarrow 1$ to E do //Extraction	
5: $[X,s1,e1] \leftarrow CCZ(N[SI+j,SI],EI); // Lower diagonal$	Step 2: Extracting the
6: $[Y,s2,e2] \leftarrow CCZ(N[SI,SI+j],EI); // Upper diagonal$	longest subsequence of
7: if $X > Y$ then $s \leftarrow s1$; $e \leftarrow e1$;	consecutive zeros
8: else $s \leftarrow s2; e \leftarrow e2;$	
9: $[X,s1,e1] \leftarrow CCZ(N[SI,SI],EI);$	
10: if $X > (e-s+1)$ then	
11: $s \leftarrow s1; e \leftarrow e1;$	
12: if (calls=1 and (e-s+1)< $(m - E)/(E + 1)$) then	Early termination condition
13: return [MAGNET, 0];	(only in first call)
14: // Right subproblem with encapsulation	Step 3: Encapsulating the
15: $[MAGNET, calls] \leftarrow EXEN(N, e+2, EI, E, MAGNET, calls+1);$	found longest subsequence
16: // Left subproblem with encapsulation	and Step 4: Divide-and-
17: [MAGNET, calls] \leftarrow EXEN(N,SI, s-2, E, MAGNET, calls+1);	Conquer Recursion
18: return [MAGNET, calls];	
19: else return [MAGNET, calls-1];	

7.2 Analysis of the MAGNET Algorithm

We analyze the asymptotic run time and space complexity of the MAGNET algorithm. MAGNET applies a divide-and-conquer technique that divides the problem of finding the common subsequences into two subproblems in each recursion call. In the first recursion call, the extracted common subsequence is of length at least a = [(m - E)/(E + 1)] bases. This reduces the problem of finding the common subsequences from *m* to at most *m*-*a*, which is further divided into two subproblems: a left subproblem and a right subproblem. For the sake of simplicity, we assume that the size of the left and the right subproblems decreases by a factor of *b* and *c*, respectively, as follows:

$$m = a + 2 + m/b + m/c$$
 (4)

The addition of 2 bases is for the encapsulation bits added at each recursion call. Now, let $T_{MAGNET}(m)$ be the time complexity of MAGNET algorithm, for identifying non-overlapping subsequences. If it takes O(km) time to find the global longest subsequence and divide the problem into two subproblems, where k = 2E+1 is the number of bit-vectors, we get the following recurrence equation:

$$T_{MAGNET}(m) = T_{MAGNET}(m/b) + T_{MAGNET}(m/c) + O(km)$$
(5)

Given that the early termination condition of MAGNET algorithm restricts the recursion depth as follows:

TMAGNET(1

Recursion tree depth =
$$\left[\log_2(E+1)\right] - 1$$
 (6)

Solving the recurrence in (5) using (4) and (6) by applying the recursion-tree method provides a loose upper-bound to the time complexity as follows:

$$n = O(km) \cdot \sum_{x=0}^{\lfloor log_2(k+1) \rfloor - 1} \left(\frac{1}{b} + \frac{1}{c}\right) \\ \approx O(fkm)$$
(7)

where *f* is a fractional number satisfies the following range: $1 \le t \le 2$. This in turn demonstrates that the MAGNET algorithm runs in linear time with respect to the sequence length and edit distance threshold and hence it is computationally inexpensive. The space complexity of the MAGNET algorithm is as follows:

$$D_{MAGNET}(m) = D_{MAGNET}(m/b) + D_{MAGNET}(m/c) + (km+m)$$

$$\approx O(fkm + fm)$$
(8)

Hence, MAGNET algorithm requires linear space with respect to the read length and edit distance threshold. Next, we describe the hardware implementation details of MAGNET filter.

7.3 Hardware Implementation

We outline the challenges that are encountered in implementing the MAGNET filter to be used in our accelerator design. Implementing the MAGNET algorithm on an FPGA is more challenging than implementing the Shouji algorithm due to the random location and variable length of each of the E+1 common subsequences. Verilog-2011 imposes two challenges on our architecture as it does not support variable-size partial selection and indexing of a group of bits from a vector (McNamara, 2001). In particular, the first challenge lies in excluding the extracted common subsequence along with its encapsulation bits from the search space of the next recursion call. The second challenge lies in dividing the problem into two subproblems, each of which has an unknown size at design time. To address these limitations and tackle the two design challenges, we keep the problem size fixed at each recursion call. We exclude the longest found subsequence from the search space by amending all bits of all 2E+1 bit-vectors that are located within the indices (locations) of the encapsulation bits to '1's. This ensures that we exclude the longest found subsequence and its corresponding location in all other bit-vectors during the subsequent recursion calls. We build the MAGNET accelerator using the same FPGA board as that used for Shouji for a fair comparison.

8 Examples of Applying the Shouji and MAGNET algorithms

In this section, we provide three examples of applying the Shouji and MAGNET filtering algorithms to different sequence pairs. In Fig. 6, we set the edit distance threshold to 4 in these examples. The diagonal vectors of the neighborhood map are horizontally presented in the same order of the diagonal vectors for a better illustration. In the first two examples (Fig. 6(a) and Fig. 6(b)), we observe that MAGNET is highly accurate in providing the exact location of the edits in the MAGNET bit-vector. This is due to two main reasons. First, MAGNET finds the exact length of each common subsequence by performing multiple individual iteration for each common subsequence. Second, it manually encapsulates each found longest subsequence of consecutive zeros by ones, which ensures to maintain the edits in the MAGNET bit-vector. On the contrary, Shouji uses overlapping search windows to detect segments of consecutive zeros. If two segments of consecutive zeros are overlapped within a single search window, then the edit between the two segments is sometimes eliminated by the overlapping zeros of the two segments as shown in Fig. 6(a).

Pairwise alignment can be performed as a *global* alignment, where two sequences of the same length are aligned end-to-end, or a *local* alignment, where subsequences of the two given sequences are aligned. It can also be performed as a *semi-global* alignment (called glocal), where the entirety of one sequence is aligned towards one of the ends of the other sequence. To ensure correct pre-alignment filtering and avoid rejecting a correct alignment, pre-alignment filter needs to consider counting the number of edits in a similar way to that of optimal alignment algorithm. This means that if the optimal alignment algorithm performs local alignment, then the pre-alignment filter should also perform local edit distance calculation. This can be achieved by not considering the leading and trailing edits in the total count of edits between two given sequences. Fig 6(a) and Fig. 6(b) show examples of global pre-alignment filtering, where the two given sequences have different lengths. While Shouji is conceptually able to perform local pre-alignment and glocal pre-alignment filtering, such support is not currently implemented in our public release of Shouji (https://github.com/CMU-SAFARI/Shouji). The current implementation of Shouji performs only global pre-alignment filtering that requires the text and reference sequences to be of the same length.



(c)

Fig. 6: Examples of applying the Shouji and MAGNET filtering algorithms to three different sequence pairs, where the edit distance threshold is set to 4. We present the content of the neighborhood map along with the Shouji and MAGNET bit-vectors. In (a) and (b), we apply Shouji and MAGNET algorithms starting from the leftmost column towards the rightmost column (end-to-end) to perform global pre-alignment filtering. In (c), we ignore the ones that are located at the two ends of the final bit-vector to perform local prealignment filtering.

9 Dataset Description

Table 5 provides the configuration used for the *-e* parameter of mrFAST (Alkan et al., 2009) for each of the 12 datasets. We use Edlib (Šošić and Šikić, 2017) to assess the number of similar (i.e., having edits fewer than or equal to the edit distance threshold) and dissimilar (i.e., having more edits than the edit distance threshold) pairs for each of the 12 datasets across different user-defined edit distance thresholds. We provide these details for set 1, set 2, set 3, and set 4 in Table 6. We provide the same details for set 5, set 6, set 7, and set 8 in Table 7 and for set 9, set 10, set 11, and set 12 in Table 8.

Table 5: Benchmark illumina-like datasets (read-reference pairs). We map each read set to the human reference genome in order to generate four datasets using different mappers' edit distance thresholds (using the *-e* parameter).

Accession no.	ERR240727_1				SRR826460_1			SRR826471_1				
Sequence Length	100				150				250			
HTS	Illumina HiSeq 2000		00	Ill	umina H	HiSeq 20	00	Il	Illumina HiSeq 2000			
Dataset	Set_1	Set_2	Set_3	Set_4	Set_5	Set_6	Set_7	Set_8	Set_9	Set_10	Set_11	Set_12
mrFAST -e	2	3	5	40	4	6	10	70	8	12	15	100
Amount of Edits	Low	-edit	High	n-edit	Low	-edit	High	n-edit	Low	v-edit	Higl	n-edit

Table 6: Details of our first four datasets (set 1, set 2, set 3, and set 4). We use Edlib to benchmark the accepted (i.e., aligned) pairs and the
rejected (i.e., unaligned) pairs for edit distance thresholds of $E=0$ up to $E=10$ edits.

Dataset	Set_1		Se	t_2	Se	t_3	Set_4		
E	Accepted	Rejected	Accepted	Rejected	Accepted	Rejected	Accepted	Rejected	
0	381,901	29,618,099	124,531	29,875,469	11,989	29,988,011	11	29,999,989	
1	1,345,842	28,654,158	441,927	29,558,073	44,565	29,955,435	18	29,999,982	
2	3,266,455	26,733,545	1,073,808	28,926,192	108,979	29,891,021	24	29,999,976	
3	5,595,596	24,404,404	2,053,181	27,946,819	206,903	29,793,097	27	29,999,973	
4	7,825,272	22,174,728	3,235,057	26,764,943	334,712	29,665,288	29	29,999,971	
5	9,821,308	20,178,692	4,481,341	25,518,659	490,670	29,509,330	34	29,999,966	
6	11,650,490	18,349,510	5,756,432	24,243,568	675,357	29,324,643	83	29,999,917	
7	13,407,801	16,592,199	7,091,373	22,908,627	891,447	29,108,553	177	29,999,823	
8	15,152,501	14,847,499	8,531,811	21,468,189	1,151,447	28,848,553	333	29,999,667	
9	16,894,680	13,105,320	10,102,726	19,897,274	1,469,996	28,530,004	711	29,999,289	
10	18,610,897	11,389,103	11,807,488	18,192,512	1,868,827	28,131,173	1,627	29,998,373	

Table 7: Details of our second four datasets (set_5, set_6, set_7, and set_8). We report the accepted and the rejected pairs for edit distance thresholds of *E*=0 up to *E*=15 edits.

Dataset	Se	t_5	Se	t_6	Se	t_7	Set_8		
E	Accepted	Rejected	Accepted	Rejected	Accepted	Rejected	Accepted	Rejected	
0	1,440,497	28,559,503	248,920	29,751,080	444	29,999,556	201	29,999,799	
1	1,868,909	28,131,091	324,056	29,675,944	695	29,999,305	327	29,999,673	
3	2,734,841	27,265,159	481,724	29,518,276	927	29,999,073	444	29,999,556	
4	3,457,975	26,542,025	612,747	29,387,253	994	29,999,006	475	29,999,525	
6	5,320,713	24,679,287	991,606	29,008,394	1,097	29,998,903	529	29,999,471	
7	6,261,628	23,738,372	1,226,695	28,773,305	1,136	29,998,864	546	29,999,454	
9	7,916,882	22,083,118	1,740,067	28,259,933	1,221	29,998,779	587	29,999,413	
10	8,658,021	21,341,979	2,009,835	27,990,165	1,274	29,998,726	612	29,999,388	
12	10,131,849	19,868,151	2,591,299	27,408,701	1,701	29,998,299	710	29,999,290	
13	10,917,472	19,082,528	2,923,699	27,076,301	2,146	29,997,854	796	29,999,204	
15	12,646,165	17,353,835	3,730,089	26,269,911	3,921	29,996,079	1,153	29,998,847	

Table 8: Details of our third four datasets (set_9, set_10, set_11, and set_12). We report the accepted and the rejected pairs for edit distance thresholds of E=0 up to E=25 edits.

Dataset	Set_9		Set	t_10	Se	t_11	Set_12		
E	Accepted	Rejected	Accepted	Rejected	Accepted	Rejected	Accepted	Rejected	
0	707,517	29,292,483	43,565	29,956,435	4,389	29,995,611	49	29,999,951	
2	1,462,242	28,537,758	88,141	29,911,859	8,970	29,991,030	163	29,999,837	
5	1,973,835	28,026,165	119,100	29,880,900	12,420	29,987,580	301	29,999,699	
7	2,361,418	27,638,582	145,290	29,854,710	15,405	29,984,595	375	29,999,625	
10	3,183,271	26,816,729	205,536	29,794,464	22,014	29,977,986	472	29,999,528	
12	3,862,776	26,137,224	257,360	29,742,640	27,817	29,972,183	520	29,999,480	
15	4,915,346	25,084,654	346,809	29,653,191	37,710	29,962,290	575	29,999,425	
17	5,550,869	24,449,131	409,978	29,590,022	44,225	29,955,775	623	29,999,377	
20	6,404,832	23,595,168	507,177	29,492,823	54,650	29,945,350	718	29,999,282	
22	6,959,616	23,040,384	572,769	29,427,231	62,255	29,937,745	842	29,999,158	
25	7,857,750	22,142,250	673,254	29,326,746	74,761	29,925,239	1,133	29,998,867	

10 Evaluating the Number of Falsely-Accepted Sequence Pairs and Falsely-Rejected Sequence Pairs

We evaluate the number of falsely-accepted pairs and falsely-rejected pairs for Shouji, MAGNET, SHD (Xin et al., 2015), and GateKeeper (Alser et al., 2017). We list the number of falsely-accepted and falsely-rejected sequences in Table 9, Table 10, and Table 11 for read lengths of 100 bp, 150 bp, and 250 bp, respectively.

The false reject rate is the ratio of the number of similar sequences that are rejected (falsely-rejected pairs) by the filter and the number of similar sequences that are accepted by the optimal sequence alignment algorithm. The false reject rate should always be equal to 0%. Using our 12 low-edit and high-edit datasets for three different sequence lengths, we observe that Shouji, SHD, and GateKeeper do *not* filter out correct sequence pairs; hence, they provide a 0% false reject rate. The reason is the way we find the common subsequences. We always look for the subsequences that have the largest number of zeros, such that we maximize the number of matches and minimize the number of edits that cause the division of one long common sequence into shorter subsequences. However, this is not the case for MAGNET. We observe that MAGNET provides a very low false reject rate of less than 0.00045% for an edit distance threshold of at least 4% of the sequence length. This is due in large part to the greedy choice of always selecting the longest common subsequences regardless of its contribution to the total number of edits. On the contrary, Shouji always examines whether or not the selected 4-bit segment that has the largest number of zeros decreases the number of edits in the Shouji bit-vector before considering the 4-bit segment to be part of the common subsequences. In Fig. 7, we show an example of where MAGNET falsely considers two given sequences as dissimilar ones, while they differ by less than the edit distance threshold. This example shows that MAGNET finds another four shorter subsequences fails in finding the two common subsequences that are highlighted in blue. Instead, MAGNET finds another four shorter subsequences that result in increasing the number of mismatches in the MAGNET bit-vector.



Fig. 7: An example of a falsely-rejected sequence pair using the MAGNET algorithm for an edit distance threshold of 6. The random zeros (highlighted in red) confuse the MAGNET filter, causing it to select shorter segments of random zeros instead of a longer common subsequences (highlighted in blue).

Table 9: Details of evaluating the number of falsely-accepted sequence pairs (FA) and falsely-rejected sequence pairs (FR) of Shouji, MAGNET, GateKeeper, and SHD using four datasets, set_1, set_2, set_3, and set_4, with a read length of 100 bp.

		Read A	ligner			Pre-a	lignr	nent Filter			
	Е	Edl	lib	SHD		GateKeeper	•	MAGNET		Shouji	
		Accepted	Rejected	FA	FR	FA	FR	FA	FR	FA	FR
	0	381,901	29,618,099	10	0	0	0	963,941	0	0	0
	1	1,345,842	28,654,158	783,185	0	783,185	0	800,099	0	333,320	0
	2	3,266,455	26,733,545	2,704,128	0	2,704,128	0	1,876,518	0	1,283,004	0
	3	5,595,596	24,404,404	5,237,529	0	5,237,529	0	2,428,301	0	2,674,876	0
÷.	4	7,825,272	22,174,728	8,231,507	0	8,231,507	0	2,662,902	1	4,399,886	0
et	5	9,821,308	20,178,692	11,195,124	0	11,195,124	0	2,916,838	0	6,452,280	0
S	6	11,650,490	18,349,510	13,781,651	0	13,781,651	0	3,406,303	4	9,373,309	0
	7	13,407,801	16,592,199	14,283,519	0	14,283,519	0	4,026,433	19	11,113,616	0
	8	15,152,501	14,847,499	13,814,295	0	13,814,295	0	4,745,672	27	11,990,529	0
	9	16,894,680	13,105,320	13,105,305	0	13,105,305	0	5,319,627	41	11,693,396	0
	10	18,610,897	11,389,103	11,389,103	0	11,389,103	0	5,673,172	31	10,664,722	0
	Ε	Accepted	Rejected	FA	FR	FA	FR	FA	FR	FA	FR
	0	124,531	29,875,469	2	0	0	0	317,396	0	0	0
	1	441,927	29,558,073	276,271	0	276,271	0	265,663	0	114,225	0
	2	1,073,808	28,926,192	1,273,787	0	1,273,787	0	779,683	0	524,886	0
	3	2,053,181	27,946,819	3,370,661	0	3,370,661	0	1,257,472	0	1,494,883	0
~	4	3,235,057	26,764,943	6,695,487	0	6,695,487	0	1,621,885	1	3,085,801	0
et	5	4,481,341	25,518,659	10,798,431	0	10,798,431	0	1,995,105	0	5,410,196	0
0,	6	5,756,432	24,243,568	15,305,752	0	15,305,752	0	2,574,171	2	9,218,900	0
	7	7,091,373	22,908,627	17,347,813	0	17,347,813	0	3,391,117	5	12,401,268	0
	8	8,531,811	21,468,189	18,015,876	0	18,015,876	0	4,485,756	19	14,865,877	0
	0	10 102 726	19 897 274	19 897 204	0	19 897 204	0	5 639 763	38	15 670 345	0
	9	10,102,720	15,057,274	10,007,201	-	19,097,201	-	3,005), 00		15,070,545	Ŭ
	9 10	11,807,488	18,192,512	18,192,512	0	18,192,512	0	6,691,920	52	15,222,777	0
	9 10 E	11,807,488 Accepted	18,192,512 Rejected	18,192,512 FA	0 FR	18,192,512 FA	0 FR	6,691,920 FA	52 FR	15,222,777 FA	0 FR
	9 10 E 0	11,807,488 Accepted 11,989	18,192,512 Rejected 29,988,011	18,192,512 FA 1	0 FR 0	18,192,512 FA 0	0 FR 0	6,691,920 FA 32,576	52 FR 0	15,010,545 15,222,777 FA 0	0 FR 0
	9 10 E 0 1	11,807,488 Accepted 11,989 44,565	19,937,274 18,192,512 Rejected 29,988,011 29,955,435	18,192,512 FA 1 30,065	0 FR 0 0	18,192,512 FA 0 30,065	0 FR 0 0	6,691,920 FA 32,576 27,639	52 FR 0 0	15,222,777 FA 0 13,060	0 FR 0 0
	9 10 E 0 1 2	10,106,720 11,807,488 Accepted 11,989 44,565 108,979	18,192,512 Rejected 29,988,011 29,955,435 29,891,021	13,192,512 FA 10,005 10,005 153,613	0 FR 0 0	18,192,512 FA 0 30,065 153,613	0 FR 0 0 0	6,691,920 FA 32,576 27,639 77,792	52 FR 0 0 0	15,010,543 15,222,777 FA 0 13,060 61,519	0 FR 0 0
	9 10 0 1 2 3	11,807,488 Accepted 11,989 44,565 108,979 206,903	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097	18,192,512 FA 1 30,065 153,613 466,411	0 FR 0 0 0 0	13,192,512 FA 0 30,065 153,613 466,411	0 FR 0 0 0 0	6,691,920 FA 32,576 27,639 77,792 133,654	52 FR 0 0 0 0 0	15,222,777 FA 0 13,060 61,519 200,269	0 FR 0 0 0
е. Г	9 10 E 0 1 2 3 4	10,102,728 11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288	18,192,512 FA 1 30,065 153,613 466,411 1,254,259	0 FR 0 0 0 0 0	18,192,512 FA 0 30,065 153,613 466,411 1,254,259	0 FR 0 0 0 0 0	6,691,920 FA 32,576 27,639 77,792 133,654 193,569	52 FR 0 0 0 0 0 0	15,070,033 15,222,777 FA 0 13,060 61,519 200,269 521,359	0 FR 0 0 0 0 0
Set_3	9 10 0 1 2 3 4 5	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674	0 FR 0 0 0 0 0 0	18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674	0 FR 0 0 0 0 0 0	6,691,920 FA 32,576 27,639 77,792 133,654 193,569 268,750	52 FR 0 0 0 0 0 0 0 0	15,222,777 FA 0 13,060 61,519 200,269 521,359 1,206,373	0 FR 0 0 0 0 0 0
Set_3	9 10 1 2 3 4 5 6	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330 29,324,643	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154	0 FR 0 0 0 0 0 0 0	18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154	0 FR 0 0 0 0 0 0 0 0	6,691,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154	52 FR 0 0 0 0 0 0 0 0 0 0 0	15,222,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331	0 FR 0 0 0 0 0 0 0 0
Set_3	9 10 E 0 1 2 3 4 5 6 7	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580	0 FR 0 0 0 0 0 0 0 0 0 0	18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580	0 FR 0 0 0 0 0 0 0 0 0 0	6,691,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154 585,853	52 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	15,222,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357	0 FR 0 0 0 0 0 0 0 0 0 0
Set_3	9 10 1 2 3 4 5 6 7 8	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	6,691,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154 585,853 931,084	52 FR 0 0 0 0 0 0 0 0 0 1 1	15,222,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786	0 FR 0 0 0 0 0 0 0 0 0 0 0 0
Set_3	9 10 0 1 2 3 4 5 6 7 8 9 10	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447 1,469,996	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553 28,530,004 29,4472	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 22,454 472	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	6,691,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154 585,853 931,084 1,466,018	52 FR 0 0 0 0 0 0 0 0 0 0 1 9 52	15,222,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786 11,228,839 12,268,276	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set_3	9 10 E 0 1 2 3 4 5 6 7 8 9 10	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447 1,469,996 1,868,827	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553 28,530,004 28,131,173 20,144	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	5,631,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154 585,853 931,084 1,466,018 2,251,403	52 FR 0 0 0 0 0 0 0 0 0 1 9 6	15,22,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786 11,228,839 13,630,704	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set_3	9 10 E 0 1 2 3 4 5 6 7 8 9 10 E	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447 1,469,996 1,868,827 Accepted	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553 28,848,553 28,530,004 28,131,173 Rejected	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	6,691,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154 585,853 931,084 1,466,018 2,251,403 FA	52 52 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	15,222,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786 11,228,839 13,630,704 FA	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0
Set_3	9 10 E 0 1 2 3 4 5 6 7 8 9 10 E 0 1	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447 1,469,996 1,868,827 Accepted 11 40	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553 28,530,004 28,131,173 Rejected 29,999,989 20,000,022	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	5,651,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154 585,853 931,084 1,466,018 2,251,403 FA 7 7	52 52 FR 0 0 0 0 0 0 0 0 0 0 0 0 1 9 6 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	15,222,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786 11,228,839 13,630,704 FA 0 2	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0
Set_3	9 10 1 2 3 4 5 6 7 8 9 10 E 0 1 1 0 1 2	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447 1,469,996 1,868,827 Accepted 11 18	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553 28,848,553 28,848,553 28,848,553 Rejected 29,999,989 29,999,989 29,999,989	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14 15 15 15 15 15 15 15 15 15 15	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	Image: Product of the system 18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	5,651,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154 585,853 931,084 1,466,018 2,251,403 FA 7 5 2 2 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3	52 52 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	15,222,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786 11,228,839 13,630,704 FA 0 2 15,222,777 0,22 0,217	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0
Set_3	9 10 1 0 1 2 3 4 5 6 7 8 9 10 E 0 1 2 3 4 5 6 7 8 9 10 E 0 1 2	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447 1,151,447 1,469,996 1,868,827 Accepted 11 18 24 24	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553 28,848,553 28,530,004 28,131,173 Rejected 29,999,989 29,999,982 29,999,976	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14 155 1106	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	Image: Product of the system 18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,529,5103 FA 0 14 1555 11406	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	6,691,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154 585,853 931,084 1,466,018 2,251,403 FA 7 5 2 2 4	52 52 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	15,22,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786 11,228,839 13,630,704 FA 0 2 15 2 2 3 16 16 16 16 16 16 16 16 16 16	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0
Set_3	9 10 1 0 1 2 3 4 5 6 7 8 9 10 10 11 2 3 4 5 6 7 8 9 10 1 2 3 4	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447 1,469,996 1,868,827 Accepted 11 18 24 24 27	18,192,512 Rejected 29,988,011 29,955,435 29,881,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553 28,848,553 28,530,004 28,131,173 Rejected 29,999,989 29,999,989 29,999,976 29,999,976 29,999,976	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14 155 1,196 7,426	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	Image: Product of the system 18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14 155 1,196 7,426	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	5,691,920 FA 32,576 27,639 77,792 133,654 193,569 92,68,750 385,154 585,853 931,084 1,466,018 2,251,403 FA 7 5 2 2 4 4 1 2 2 4 1 2 2 2 4 1 2 2 2 2 2 4 1 2 2 2 2 2 2 2 2 2 2 2 2 2	52 52 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	15,22,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786 11,228,839 13,630,704 FA 0 2 15 216 15 216 15 216 15 15 216 15 15 216 15 15 15 15 15 15 15 15 15 15	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0
t_4 Set_3	10 10 1 2 3 4 5 6 7 8 9 10 10 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447 1,469,996 1,868,827 Accepted 11 18 24 27 29 24	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553 28,848,553 28,530,004 28,131,173 Rejected 29,999,989 29,999,989 29,999,976 29,999,971 29,999,971 29,999,971	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14 155 1,196 7,436 27,02	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	Image: 18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14 155 1,196 7,436	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	5,691,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154 585,853 931,084 1,466,018 2,251,403 FA 7 5 2 2 4 13 3 2 2 2 4 13 3 2 2 2 2 2 2 2 2 2 2 2 2 2	52 52 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	15,22,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786 11,228,839 13,630,704 FA 0 2 15 216 1,986 10,955	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0
Set_4 Set_3	9 10 6 7 8 9 10 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447 1,469,996 1,868,827 Accepted 11 18 24 27 29 34	29,957,272 Rejected 29,988,011 29,955,435 29,881,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553 28,848,553 28,530,004 28,131,173 Rejected 29,999,982 29,999,982 29,999,973 29,999,971 29,999,971 29,999,971	18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14 155 1,196 7,436 32,792 155 12,5124	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	Image: Construct of the system 18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14 155 1,196 7,436 32,792 155	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	5,691,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154 585,853 931,084 1,466,018 2,251,403 FA 7 7 5 2 2 4 13 82 2 209	52 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	15,22,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786 11,228,839 13,630,704 FA 0 2 15 216 1,985 10,955 57,259	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0
Set_4 Set_3	9 10 0 1 2 3 4 5 6 7 8 9 10 2 3 4 5 6 7 8 9 10 E 0 1 2 3 4 5 6 7	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447 1,469,996 1,868,827 Accepted 11 18 24 24 27 29 34 83	29,957,272 Rejected 29,988,011 29,955,435 29,881,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553 28,530,004 28,131,173 Rejected 29,999,982 29,999,982 29,999,971 29,999,971 29,999,917 29,999,917	Image: Constraint of the system 18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14 155 1,196 7,436 32,792 155,134	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	Image: Product of the system 18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 114 155 1,196 7,436 32,792 155,134	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	5,691,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154 585,853 931,084 1,466,018 2,251,403 FA 7 7 5 2 2 4 133 82 298 1022	52 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	15,22,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786 11,228,839 13,630,704 FA 0 2 15 216 1,986 10,551 57,258 214.005	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0
Set_4 Set_3	9 10 E 0 1 2 3 4 5 6 7 8 9 10 E 0 1 2 3 4 5 6 7 8 9 10 E 0 1 2 3 4 5 6 7 8	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447 1,469,996 1,868,827 Accepted 11 18 24 24 27 29 34 83 177 29 34	18,192,512 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553 28,530,004 28,131,173 Rejected 29,999,989 29,999,989 29,999,982 29,999,976 29,999,971 29,999,971 29,999,966 29,999,917 29,999,966 29,999,917 29,999,923 29,999,921	Image: Constraint of the system 18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14 155 1,196 7,436 32,792 155,134 417,444 1031,420	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	Image: Construct of the system 18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,627,154 9,695,580 12,921,874 28,529,540 28,131,173 FA 0 14 155 1,196 7,436 32,792 155,134 417,444 1031,420	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	5,691,920 FA 32,576 27,639 77,792 133,654 193,569 268,750 385,154 585,853 931,084 1,466,018 2,251,403 FA 7 7 5 2 2 4 133 82 298 1,030 3 120	52 52 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	15,22,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786 11,228,839 13,630,704 FA 0 2 11,228,839 13,630,704 FA 0 2 15 216 1,986 10,551 57,258 214,005 675,029	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0
Set_4 Set_3	9 10 E 0 1 2 3 4 5 6 7 8 9 10 2 3 4 5 6 7 8 9 10 2 3 4 5 6 7 8 9	11,807,488 Accepted 11,989 44,565 108,979 206,903 334,712 490,670 675,357 891,447 1,151,447 1,469,996 1,868,827 Accepted 11 18 24 27 29 34 83 177 333 711	29,957,12 Rejected 29,988,011 29,955,435 29,891,021 29,793,097 29,665,288 29,509,330 29,324,643 29,108,553 28,848,553 28,848,553 28,848,553 28,848,553 28,848,553 28,939,004 28,131,173 Rejected 29,999,982 29,999,982 29,999,971 29,999,966 29,999,917 29,999,823 29,999,824	Image: Constraint of the system 18,192,512 FA 1 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,874 28,529,540 28,529,540 28,131,173 FA 0 14 1,196 7,436 32,792 155,134 417,444 1,031,480 29,997,022	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	Image: Product of the system 18,192,512 FA 0 30,065 153,613 466,411 1,254,259 2,767,674 6,227,154 9,695,580 12,921,887 28,529,540 28,131,173 FA 0 14 155,134 417,444 1,031,480 29,97022	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	5,651,920 FA 32,576 27,639 77,792 133,654 133,569 268,750 385,154 585,853 931,084 1,466,018 2,251,403 FA 7 5 2 2 4 133 82 298 1,030 3,129 8,234	52 52 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	15,222,777 FA 0 13,060 61,519 200,269 521,359 1,206,373 2,983,331 5,431,357 8,532,786 11,228,839 13,630,704 FA 0 2 13,630,704 FA 0 2 13,551 57,258 214,005 675,029 1,742,476	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0

Table 10: Details of evaluating the number of falsely-accepted sequence pairs (FA) and falsely-rejected sequence pairs (FR) of Shouj
MAGNET, GateKeeper, and SHD using four datasets, set_5, set_6, set_7, and set_8, with a read length of 150 bp.

		Read A	Aligner			Pre-al						
	Е	Ed	lib	SHD		GateKeep	er	MAGNE	т	Shouji		
		Accepted	Rejected	FA	FR	FA	FR	FA	FR	FA	FR	
	0	1,440,497	28,559,503	0	0	0	0	428,412	0	0	0	
	1	1,868,909	28,131,091	173,573	0	173,573	0	156,891	0	113,519	0	
	3	2,734,841	27,265,159	2,080,279	0	2,080,279	0	725,873	0	1,539,365	0	
	4	3,457,975	26,542,025	4,023,762	0	4,023,762	0	1,064,344	0	3,042,831	0	
ın.	6	5,320,713	24,679,287	9,258,602	0	9,258,602	0	1,430,272	0	6,025,592	0	
et	7	6,261,628	23,738,372	12,481,853	0	12,481,853	0	1,532,024	2	8,219,336	0	
S	9	7,916,882	22,083,118	22,076,837	0	22,076,837	0	1,874,734	20	14,568,337	0	
	10	8,658,021	21,341,979	21,341,979	0	21,341,979	0	2,194,275	10	16,920,389	0	
	12	10,131,849	19,868,151	19,868,151	0	19,868,151	0	3,294,672	42	18,270,597	0	
	13	10,917,472	19,082,528	19,082,528	0	19,082,528	0	4,066,617	46	18,095,207	0	
	15	12,646,165	17,353,835	17,353,835	0	17,353,835	0	5,810,797	62	16,993,568	0	
	Е	Accepted	Rejected	FA	FR	FA	FR	FA	FR	FA	FR	
	0	248,920	29,751,080	0	0	0	0	75,136	0	0	0	
	1	324,056	29,675,944	31,406	0	31,406	0	28,456	0	20,294	0	
	3	481,724	29,518,276	440,577	0	440,577	0	131,460	0	309,015	0	
	4	612,747	29,387,253	1,023,901	0	1,023,901	0	199,248	0	718,847	0	
و	6	991,606	29,008,394	4,165,422	0	4,165,422	0	334,729	0	2,222,934	0	
Set	7	1,226,695	28,773,305	7,137,889	0	7,137,889	0	405,052	0	3,762,706	0	
•••	9	1,740,067	28,259,933	28,215,257	0	28,215,257	0	600,124	0	10,299,935	0	
	10	2,009,835	27,990,165	27,990,165	0	27,990,165	0	753,866	2	13,826,393	0	
	12	2,591,299	27,408,701	27,408,701	0	27,408,701	0	1,336,246	10	17,542,652	0	
	17	2 022 600	27 076 201	27 076 301	0	27 076 301	0	1 835 77/	10	10 271 EC2	0	
	13	2,923,099	27,070,301	27,070,301	0	27,070,301		1,035,774	15	10,371,303	0	
	15	3,730,089	26,269,911	26,269,911	0	26,269,911	0	3,354,276	33	19,528,254	0	
	15 15 E	3,730,089 Accepted	26,269,911 Rejected	26,269,911 FA	0 FR	26,269,911 FA	0 FR	3,354,276 FA	33 FR	19,528,254 FA	0 FR	
	13 15 E 0	3,730,089 Accepted 444	26,269,911 Rejected 29,999,556	26,269,911 FA 0	0 FR 0	26,269,911 FA	0 FR 0	3,354,276 FA 251	33 FR 0	19,528,254 FA	0 FR 0	
	15 15 E 0 1	2,923,099 3,730,089 Accepted 444 695	26,269,911 Rejected 29,999,556 29,999,305	26,269,911 FA 0 104	0 FR 0 0	26,269,911 FA 0 104	0 FR 0 0	3,354,276 FA 251 77	33 FR 0 0	19,528,254 FA 0 94	0 FR 0 0	
	13 15 E 0 1 3	2,923,099 3,730,089 Accepted 444 695 927	26,269,911 Rejected 29,999,556 29,999,305 29,999,073	26,269,911 FA 0 104 191	0 FR 0 0 0	26,269,911 FA 0 104 191	0 FR 0 0	1,033,774 3,354,276 FA 251 77 68	13 33 FR 0 0 0	18,371,363 19,528,254 FA 0 94 180	0 FR 0 0	
	15 15 0 1 3 4	2,923,039 3,730,089 Accepted 444 695 927 994	26,269,911 Rejected 29,999,556 29,999,305 29,999,006 29,999,006	26,269,911 FA 0 104 191 643	0 FR 0 0 0 0	26,269,911 FA 0 104 191 643	0 FR 0 0 0	3,354,276 FA 251 77 68 53	33 FR 0 0 0	16,371,363 19,528,254 FA 0 94 180 421	0 FR 0 0 0 0 0	
_7	13 15 0 1 3 4 6	2,923,099 3,730,089 Accepted 444 695 927 994 1,097	27,077,301 26,269,911 Rejected 29,999,305 29,999,005 29,999,006 29,999,006	26,269,911 26,269,911 FA 0 104 191 643 47,924	0 FR 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924	0 FR 0 0 0 0 0 0	3,354,276 FA 251 77 68 53 57 77	13 33 FR 0 0 0 0 0	19,528,254 19,528,254 FA 0 94 180 421 19,097 70,540	0 FR 0 0 0 0 0	
Set_7	13 15 0 1 3 4 6 7	2,923,099 3,730,089 Accepted 444 695 927 994 1,097 1,136	27,077,301 26,269,911 Rejected 29,999,305 29,999,305 29,999,006 29,998,903 29,998,864 29,999,375	26,269,911 FA 0 104 191 643 47,924 175,481 20,555,245	0 FR 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 104 191 643 47,924 175,481 20,555,255	0 FR 0 0 0 0 0	1,033,774 3,354,276 FA 251 77 68 53 57 74	I I 33 FR 0 0 0 0 0 0 0 0	19,528,254 19,528,254 FA 0 94 180 421 19,097 70,540	0 0 0 0 0 0 0 0	
Set_7	13 15 0 1 3 4 6 7 9	3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221	26,269,911 Rejected 29,999,556 29,999,005 29,999,006 29,998,903 29,998,864 29,998,864 29,998,779 29,998,779	26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 20,000 355	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 20,000 325	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	3,354,276 FA 251 777 68 53 57 74 461 401	13 33 FR 0 0 0 0 0 0 0 0 0 0	19,528,254 19,528,254 FA 0 94 180 421 19,097 70,540 857,547 1 9222	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
Set_7	13 15 0 1 3 4 6 7 9 10	2,322,039 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274	26,269,911 Rejected 29,999,556 29,999,305 29,999,006 29,999,006 29,998,903 29,998,864 29,998,779 29,998,726	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,598,726 29,299,726	0 FR 0 0 0 0 0 0 0 0 0 0	3,354,276 FA 251 777 688 533 577 744 461 1,017 4,214	13 33 FR 0 0 0 0 0 0 0 0 0 0 0 0	18,57,1,363 19,528,254 FA 0 94 180 421 19,097 70,540 857,547 1,829,338 4,292,328	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
Set_7	13 15 E 0 1 3 4 6 7 9 10 12	2,322,039 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,274	26,269,911 Rejected 29,999,556 29,999,005 29,999,006 29,998,903 29,998,864 29,998,779 29,998,726 29,998,726 29,998,726	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,229 29,998,229	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,796 29,998,299 29,998,295	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	3,354,276 FA 251 777 688 533 577 74 461 1,017 4,218 0,628	IJ II 33 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	18,57,1,505 19,528,254 FA 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 C 055,205	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
Set_7	13 15 0 1 3 4 6 7 9 10 12 13	2,322,033 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,274 1,701 2,146	26,269,911 Rejected 29,999,556 29,999,005 29,999,005 29,999,006 29,998,903 29,998,864 29,998,779 29,998,726 29,998,726 29,998,726 29,997,854 29,997,854	26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,726 29,997,854 29,997,854	6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,299 29,997,854	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	3,354,276 FA 251 77 68 53 57 74 461 1,017 4,218 8,620 21 762	IJ I 33 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	19,571,503 19,528,254 FA 0 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 1,955,205	0 0 0 0 0 FR 0 0 0 0 0 0 0 0 0 0 0	
Set_7	13 15 0 1 3 4 6 7 9 10 12 13 15 15 10 12 13 15 </th <th>2,322,033 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,701 2,146 3,921</th> <th>26,269,911 Rejected 29,999,556 29,999,005 29,999,005 29,999,006 29,998,903 29,998,864 29,998,779 29,998,726 29,998,726 29,998,299 29,997,854 29,997,854 29,996,079 Decement</th> <th>26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,299 29,997,854 29,996,079 52</th> <th>6 0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</th> <th>26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,299 29,997,854 29,996,079</th> <th>0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0</th> <th>3,354,276 FA 251 77 68 53 57 74 461 1,017 4,218 8,620 31,783</th> <th>13 33 FR 0</th> <th>19,571,303 19,528,254 FA 0 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488</th> <th>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</th>	2,322,033 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,701 2,146 3,921	26,269,911 Rejected 29,999,556 29,999,005 29,999,005 29,999,006 29,998,903 29,998,864 29,998,779 29,998,726 29,998,726 29,998,299 29,997,854 29,997,854 29,996,079 Decement	26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,299 29,997,854 29,996,079 5 2	6 0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,299 29,997,854 29,996,079	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	3,354,276 FA 251 77 68 53 57 74 461 1,017 4,218 8,620 31,783	13 33 FR 0	19,571,303 19,528,254 FA 0 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
Set_7	13 15 1 15 1 3 4 6 7 9 10 12 13 15 <td>2,223,039 3,730,089 Accepted 444 695 9927 994 1,097 1,136 1,221 1,274 1,701 2,146 3,921 Accepted</td> <td>26,269,911 Rejected 29,999,556 29,999,005 29,999,005 29,999,006 29,998,864 29,998,779 29,998,726 29,998,726 29,998,299 29,997,854 29,997,854 29,996,079 Rejected</td> <td>26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,299 29,997,854 29,996,079 FA</td> <td>0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0</td> <td>26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,299 29,997,854 29,996,079 FA</td> <td>0 FR 0</td> <td>3,354,276 FA 251 77 68 53 57 74 461 1,017 4,218 8,620 31,783 FA</td> <td>33 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td> <td>19,571,303 19,528,254 FA 0 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA</td> <td>0 <mark>FR</mark> 0 0 0 0 0 0 0 0 0 0 FR 0</td>	2,223,039 3,730,089 Accepted 444 695 9927 994 1,097 1,136 1,221 1,274 1,701 2,146 3,921 Accepted	26,269,911 Rejected 29,999,556 29,999,005 29,999,005 29,999,006 29,998,864 29,998,779 29,998,726 29,998,726 29,998,299 29,997,854 29,997,854 29,996,079 Rejected	26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,299 29,997,854 29,996,079 FA	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,299 29,997,854 29,996,079 FA	0 FR 0	3,354,276 FA 251 77 68 53 57 74 461 1,017 4,218 8,620 31,783 FA	33 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	19,571,303 19,528,254 FA 0 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA	0 <mark>FR</mark> 0 0 0 0 0 0 0 0 0 0 FR 0	
Set_7	13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1	2,223,039 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,274 1,701 2,146 3,921 Accepted 201	26,269,911 Rejected 29,999,556 29,999,305 29,999,005 29,999,006 29,998,864 29,998,709 29,998,726 29,998,726 29,998,726 29,998,729 29,998,726 29,998,729 29,997,854 29,997,854 29,999,799 Rejected 29,999,799	26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,997,854 29,996,079 FA 0 0	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,729 29,997,854 29,996,079 FA 0 0	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	3,354,276 FA 251 777 688 533 577 744 461 1,017 4,218 8,620 31,783 FA 126	33 FR 0	19,571,303 19,528,254 FA 0 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
Set_7	13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12 13 12 12 12 12 12 12 12 12 12 12 12 12 12	2,223,039 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,274 1,701 2,146 3,921 Accepted 201 327	29,999,305 29,999,305 29,999,305 29,999,006 29,999,006 29,998,003 29,998,864 29,998,709 29,998,726 29,998,726 29,998,729 29,998,726 29,998,729 29,997,854 29,997,854 29,999,673 29,999,556	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,997,854 29,996,079 FA 0 58 90	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,729 29,997,854 29,996,079 FA 0 58 90	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	3,354,276 FA 251 777 68 53 57 74 461 1,017 4,218 8,620 31,783 FA 126 42 25	33 FR 0	19,571,303 19,528,254 FA 0 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA 0 43 83 83 83 83 83 83 83 83 83 8	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
Set_7	13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12 13 10 1 3 4 6 7 9 10 10 10 10 10 10 10 10 10 10 10 10 10	2,223,039 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,274 1,701 2,146 3,921 Accepted 201 327 327 444	29,999,305 29,999,305 29,999,305 29,999,006 29,999,006 29,998,003 29,998,864 29,998,709 29,998,726 29,998,726 29,998,726 29,998,729 29,997,854 29,997,854 29,999,799 29,999,673 29,999,556 29,999,556	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,729 29,997,854 29,996,079 FA 0 58 90 267	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,729 29,997,854 29,996,079 FA 0 58 90 90	6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	3,354,276 FA 251 777 68 53 57 74 461 1,017 4,218 8,620 31,783 FA 126 42 35 72 72 74 74 74 75 76 76 77 76 76 77 76 77 76 77 76 77 77	Image: 100 minipage Image: 100 minipage <thimage: 100="" minipage<="" th=""> Image: 100 minipage</thimage:>	19,57,303 19,528,254 FA 0 0 4 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA 0 43 83 83 137	0 0	
Set_7	13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1 3 4 6	2,222,033 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,701 2,146 3,921 Accepted 201 327 444 475 520	2,907,500 26,269,911 Rejected 29,999,305 29,999,003 29,999,006 29,998,903 29,998,804 29,998,709 29,998,709 29,998,729 29,998,729 29,998,299 29,997,854 29,999,785 29,999,799 29,999,735 29,999,735 29,999,525 29,999,525	27,070,301 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,229 29,997,854 29,998,229 EA 0 FA 0 58 900 267 18110	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,998,726 29,998,726 29,998,726 29,998,729 29,997,854 29,996,079 FA 0 588 900 2677 18,110	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	3,354,276 FA 251 777 68 53 57 74 461 1,017 4,218 8,620 31,783 FA 126 42 35 28 28 28	33 FR 0	18,371,363 19,528,254 FA 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA 0 43 83 137 6,259	0 0	
t_8 Set_7	13 14 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1 3 4 6 7	2,222,033 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,701 2,146 3,921 Accepted 201 3,27 444 475 529 524	26,269,911 Rejected 29,999,305 29,999,003 29,999,003 29,999,006 29,998,903 29,998,903 29,998,709 29,998,729 29,998,729 29,998,729 Rejected 29,999,7854 29,999,799 29,999,673 29,999,673 29,999,454	27,070,301 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,229 29,997,854 29,998,229 EA 0 58 90 267 18,110 79 419	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,995,345 29,998,726 29,998,726 29,998,729 29 ,997,854 29,996,079 FA 0 58 900 267 18,110 79,419	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	3,354,276 FA 251 777 68 53 57 74 461 1,017 4,218 8,620 31,783 FA 126 42 35 28 28 27	33 FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	18,371,363 19,528,254 FA 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA 0 4,33 83 137 6,259 27,092 27,092 70,920 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
Set_8 Set_7	13 15 E 0 1 3 4 6 7 9 10 12 13 13 4 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9	2,223,039 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,274 1,274 2,146 3,921 Accepted 201 327 444 475 529 546 587	2,999,305 29,999,305 29,999,003 29,999,003 29,999,006 29,998,903 29,998,864 29,998,709 29,998,709 29,998,729 29,998,299 29,997,854 29,999,785 29,999,799 29,999,607 Rejected 29,999,799 29,999,673 29,999,556 29,999,556 29,999,471 29,999,413	27,070,301 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,997,854 29,997,854 29,996,079 FA 0 58 90 267 18,110 79,418 29,668,666	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	24,96,931 26,269,911 FA 0 104 191 643 47,924 175,481 29,995,345 29,998,726 29,998,726 29,998,726 29,998,729 29,997,854 29,996,079 FA 0 58 90 267 18,110 79,418 29,688,666	6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7	3,354,276 FA 251 777 688 533 577 744 461 1,017 4,218 8,620 31,783 FA 126 422 335 28 25 27 108	33 FR 0	18,371,363 19,528,254 FA 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA 0 4,33 137 6,259 27,092 27,092 20,474 20,292 20,292 20,474 20,292	0 0	
Set_8 Set_7	13 15 16 7 9 10 12 13 15 16 7 9 10 12 13 15 15 15 16 7 9 10	2,223,039 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,274 1,274 2,146 3,921 Accepted 201 327 444 475 529 546 587 612	2,999,305 29,999,556 29,999,305 29,999,006 29,999,006 29,998,903 29,998,864 29,998,709 29,998,779 29,998,779 29,998,726 29,998,726 29,997,854 29,996,079 Rejected 29,999,785 29,999,785 29,999,785 29,999,785 29,999,795 29,999,795 29,999,785 20,999,785 20,999,795 20,999,795 20,995 20,995 20,995 20,995 20,995 20,995 20,995 20,995 20,995 20,9	27,070,301 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,726 29,997,854 29,996,079 FA 0 58 90 267 18,110 79,418 29,698,666 29,999,388	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,995,345 29,998,726 29,998,726 29,998,729 29,997,854 29,996,079 FA 0 58 90 26,7 18,110 79,418 29,698,666	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	3,354,276 FA 251 777 68 53 57 74 461 1,017 4,218 8,620 31,783 FA 126 42 35 28 25 27 108 231	Image: 100 minipage Image: 100 minipage <thimage: 100="" minipage<="" th=""> Image: 100 minipage</thimage:>	19,57,303 19,528,254 FA 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA 0 4,833,299 6,955,205 12,854,488 FA 0 4,833,295 27,092 27,092 404,742 935,486	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
Set_8 Set_7	13 1 6 7 9 10 12 13 4 6 7 9 10 12 13 14 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12	2,223,039 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,274 1,274 2,146 3,921 Accepted 201 327 444 475 529 546 587 587 612 710	26,269,911 Rejected 29,999,556 29,999,003 29,999,006 29,998,003 29,998,003 29,998,709 29,998,709 29,998,726 29,998,726 29,998,726 29,997,854 29,996,079 Rejected 29,999,785 20,999,785 20,995 20,995 20,995 20,995 20,995 20,995 20,995 20,995 20	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,726 29,998,726 29,997,854 29,996,079 FA 0 58 90 267 18,110 79,418 29,698,666 29,999,388 29,999,388 29,999,290	0 FR 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,726 29,998,726 0 FA 0 FA 0 58 90 26,77 18,110 79,418 29,698,666 29,999,388 29,999,388 29,999,284	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	3,354,276 FA 251 777 68 53 57 74 461 1,017 4,218 8,620 31,783 FA 126 42 35 28 25 27 108 231 965	Image: 100 minipage Image: 100 minipage <thimage: 100="" minipage<="" th=""> Image: 100 minipage</thimage:>	19,57,303 19,528,254 FA 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA 0 43 83 137 6,259 27,092 404,742 935,486 2 514 950	0 R 0	
Set_8 Set_7	13 14 6 7 9 10 12 13 14 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12 13 15 E 0 1 3 4 6 7 9 10 12 13 1 1 3 4 6 7 9 10 12 13 1 3 4 6 7 9 10 13 4 6 7 9 10 <td co<="" td=""><td>2,223,039 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,274 2,146 3,921 Accepted 201 327 444 475 529 546 587 612 587 612 710 796</td><td>26,269,911 Rejected 29,999,556 29,999,305 29,999,003 29,999,004 29,998,003 29,998,864 29,998,709 29,998,709 29,998,726 29,998,726 29,997,854 29,997,854 29,997,854 29,997,854 29,999,709 Rejected 29,999,709 29,999,704 29,999,704 29,999,404 29,999,204</td><td>26,09,001 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,726 29,998,729 FA 0 FA 0 58 90 267 18,110 79,418 29,698,666 29,999,388 29,999,204</td><td>0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0</td><td>26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,726 29,998,726 29,998,726 29,998,726 0 FA 0 FA 0 FA 0 58 900 26,7 18,110 79,418 29,698,666 29,999,388 29,999,290 29,999,204</td><td>0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0</td><td>3,354,276 FA 251 777 68 53 57 74 461 1,017 4,218 8,620 31,783 FA 126 42 355 225 27 108 231 965 2,018</td><td>Image: height of the second second</td><td>19,57,303 19,528,254 FA 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA 0 4,33 6,255 27,092 404,742 935,446 2,514,950 3,693,298</td><td>0 R 0</td></td>	<td>2,223,039 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,274 2,146 3,921 Accepted 201 327 444 475 529 546 587 612 587 612 710 796</td> <td>26,269,911 Rejected 29,999,556 29,999,305 29,999,003 29,999,004 29,998,003 29,998,864 29,998,709 29,998,709 29,998,726 29,998,726 29,997,854 29,997,854 29,997,854 29,997,854 29,999,709 Rejected 29,999,709 29,999,704 29,999,704 29,999,404 29,999,204</td> <td>26,09,001 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,726 29,998,729 FA 0 FA 0 58 90 267 18,110 79,418 29,698,666 29,999,388 29,999,204</td> <td>0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0</td> <td>26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,726 29,998,726 29,998,726 29,998,726 0 FA 0 FA 0 FA 0 58 900 26,7 18,110 79,418 29,698,666 29,999,388 29,999,290 29,999,204</td> <td>0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0</td> <td>3,354,276 FA 251 777 68 53 57 74 461 1,017 4,218 8,620 31,783 FA 126 42 355 225 27 108 231 965 2,018</td> <td>Image: height of the second second</td> <td>19,57,303 19,528,254 FA 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA 0 4,33 6,255 27,092 404,742 935,446 2,514,950 3,693,298</td> <td>0 R 0</td>	2,223,039 3,730,089 Accepted 444 695 927 994 1,097 1,136 1,221 1,274 1,274 2,146 3,921 Accepted 201 327 444 475 529 546 587 612 587 612 710 796	26,269,911 Rejected 29,999,556 29,999,305 29,999,003 29,999,004 29,998,003 29,998,864 29,998,709 29,998,709 29,998,726 29,998,726 29,997,854 29,997,854 29,997,854 29,997,854 29,999,709 Rejected 29,999,709 29,999,704 29,999,704 29,999,404 29,999,204	26,09,001 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,726 29,998,729 FA 0 FA 0 58 90 267 18,110 79,418 29,698,666 29,999,388 29,999,204	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	26,269,911 26,269,911 FA 0 104 191 643 47,924 175,481 29,595,345 29,998,726 29,998,726 29,998,726 29,998,726 29,998,726 29,998,726 0 FA 0 FA 0 FA 0 58 900 26,7 18,110 79,418 29,698,666 29,999,388 29,999,290 29,999,204	0 FR 0 0 0 0 0 0 0 0 0 0 0 0 0	3,354,276 FA 251 777 68 53 57 74 461 1,017 4,218 8,620 31,783 FA 126 42 355 225 27 108 231 965 2,018	Image: height of the second	19,57,303 19,528,254 FA 0 94 180 421 19,097 70,540 857,547 1,829,338 4,893,299 6,955,205 12,854,488 FA 0 4,33 6,255 27,092 404,742 935,446 2,514,950 3,693,298	0 R 0

Table 11: Details of evaluating the number of falsely-accepted sequence pairs (FA) and falsely-rejected sequence pairs (FR) of Shouji, MAGNET, GateKeeper, and SHD using four datasets, set_9, set_10, set_11, and set_12, with a read length of 250 bp.

		Read	Aligner			Pre-alignment Filter					
	Е	Ed	llib	SHD		GateKeep	er	MAGNE	т	Shouji	
		Accepted	Rejected	FA	FR	FA	FR	FA	FR	FA	FR
	0	707,517	29,292,483	0	0	0	0	479,104	0	0	0
	2	1,462,242	28,537,758	238,368	0	238,368	0	143,066	0	174,366	0
	5	1,973,835	28,026,165	1,546,126	0	1,546,126	0	226,864	0	1,071,218	0
	7	2,361,418	27,638,582	3,933,916	0	3,933,916	0	347,819	1	2,775,419	0
<u>ە</u>	10	3,183,271	26,816,729	26,816,729	0	26,816,729	0	624,927	1	6,669,084	0
et	12	3,862,776	26,137,224	26,137,224	0	26,137,224	0	825,468	9	11,147,373	0
5	15	4,915,346	25,084,654	25,084,654	0	25,084,654	0	1,066,633	14	18,406,823	0
	17	5,550,869	24,449,131	24,449,131	0	24,449,131	0	1,235,999	23	20,971,826	0
	20	6,404,832	23,595,168	23,595,168	0	23,595,168	0	1,695,351	35	22,223,170	0
	22	6,959,616	23,040,384	23,040,384	0	23,040,384	0	2,241,984	42	22,271,215	0
	25	7,857,750	22,142,250	22,142,250	0	22,142,250	0	3,514,515	54	21,849,454	0
	Ε	Accepted	Rejected	FA	FR	FA	FR	FA	FR	FA	FR
	0	43,565	29,956,435	0	0	0	0	28,540	0	0	0
	2	88,141	29,911,859	13,092	0	13,092	0	8,367	0	11,238	0
	5	119,100	29,880,900	113,106	0	113,106	0	14,685	0	77,095	0
	7	145,290	29,854,710	364,611	0	364,611	0	24,919	0	227,073	0
17	10	205,536	29,794,464	29,794,464	0	29,794,464	0	45,768	0	/82,844	0
Set	12	257,360	29,742,640	29,742,640	0	29,742,640	0	63,557	2	2,195,021	0
	15	346,809	29,653,191	29,653,191	0	29,653,191	0	92,443	1	7,573,911	0
	17	409,978	29,590,022	29,590,022	0	29,590,022	0	116,740	1	11,603,069	0
	20	507,177	29,492,823	29,492,823	0	29,492,823	0	217,274	2	10,075,487	0
	22	572,769	29,427,231	29,427,231	0	29,427,231	0	217,274	5	19,167,498	0
	25	673,254	29,326,746	29,326,746	0	29,326,746	0	376,323	/	24,778,497	0
	-	Accontod	Paiastad	EA	ED	E۸	ED	E A	ED	EA	ED
	Е	Accepted	Rejected	FA 0	FR 0	FA	FR	FA 2 933	FR 0	FA	FR 0
	Е 0 2	Accepted 4,389 8,970	Rejected 29,995,611 29,991,030	FA 0 1.405	FR 0	FA 0 1.405	FR 0	FA 2,933 890	FR 0	FA 0 1.173	FR 0 0
	E 0 2 5	Accepted 4,389 8,970 12,420	Rejected 29,995,611 29,991,030 29,987,580	FA 0 1,405 12,185	FR 0 0	FA 0 1,405 12,185	FR 0 0	FA 2,933 890 1.704	FR 0 0	FA 0 1,173 8,489	FR 0 0
	E 0 2 5 7	Accepted 4,389 8,970 12,420 15,405	Rejected 29,995,611 29,991,030 29,987,580 29,984,595	FA 0 1,405 12,185 41,555	FR 0 0 0	FA 0 1,405 12,185 41.555	FR 0 0 0	FA 2,933 890 1,704 2,644	FR 0 0 0	FA 0 1,173 8,489 24,946	FR 0 0 0 0
1	E 0 2 5 7 10	Accepted 4,389 8,970 12,420 15,405 22,014	Rejected 29,995,611 29,991,030 29,987,580 29,984,595 29,977,986	FA 0 1,405 12,185 41,555 29,977,986	FR 0 0 0 0 0	FA 0 1,405 12,185 41,555 29,977,986	FR 0 0 0 0 0	FA 2,933 890 1,704 2,644 4,759	FR 0 0 0 0 0	FA 0 1,173 8,489 24,946 145.053	FR 0 0 0 0
t_11	E 0 2 5 7 10 12	Accepted 4,389 8,970 12,420 15,405 22,014 27,817	Rejected 29,995,611 29,991,030 29,987,580 29,984,595 29,977,986 29,972,183	FA 0 1,405 12,185 41,555 29,977,986 29,972,183	FR 0 0 0 0 0 0	FA 0 1,405 12,185 41,555 29,977,986 29,972,183	FR 0 0 0 0 0 0	FA 2,933 890 1,704 2,644 4,759 6,729	FR 0 0 0 0 1	FA 0 1,173 8,489 24,946 145,053 833,703	FR 0 0 0 0 0 0 0 0 0
Set_11	E 0 2 5 7 10 12 15	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710	Rejected 29,995,611 29,991,030 29,987,580 29,984,595 29,977,986 29,972,183 29,962,290	FA 0 1,405 12,185 41,555 29,977,986 29,972,183 29,962,290	FR 0 0 0 0 0 0 0	FA 0 1,405 12,185 41,555 29,977,986 29,972,183 29,962,290	FR 0 0 0 0 0 0 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498	FR 0 0 0 0 1 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387	FR 0 0 0 0 0 0 0 0
Set_11	E 0 2 5 7 10 12 15 17	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225	Rejected 29,995,611 29,991,030 29,987,580 29,984,595 29,977,986 29,972,183 29,962,290 29,955,775	FA 0 1,405 12,185 41,555 29,977,986 29,972,183 29,962,290 29,955,775	FR 0 0 0 0 0 0 0 0 0 0	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775	FR 0 0 0 0 0 0 0 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134	FR 0 0 0 0 0 1 0 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285	FR 0
Set_11	E 0 2 5 7 10 12 15 17 20	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650	Rejected 29,995,611 29,991,030 29,987,580 29,984,595 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350	FA 0 1,405 12,185 41,555 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350	FR 0 0 0 0 0 0 0 0 0 0	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350	FR 0 0 0 0 0 0 0 0 0 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366	FR 0 0 0 0 0 1 0 0 0 0 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067	FR 0
Set_11	E 0 2 5 7 10 12 15 17 20 22	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 62,255	Rejected 29,995,611 29,991,030 29,987,580 29,984,595 29,972,183 29,962,290 29,955,775 29,945,350 29,945,350	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,937,745	FR 0 0 0 0 0 0 0 0 0 0 0 0	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,945,350	FR 0 0 0 0 0 0 0 0 0 0 0 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366 25,411	FR 0 0 0 0 1 0 0 0 0 2	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067 20,798,178	FR 0
Set_11	E 0 2 5 7 100 122 155 177 200 222 255	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 62,255 74,761	Rejected 29,995,611 29,991,030 29,987,580 29,984,595 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,925,239	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,937,745 29,925,239	FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,937,745 29,925,239	FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366 25,411 44,377	FR 0 0 0 0 0 1 0 0 0 0 2 1	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067 20,798,178 26,094,659	FR 0
Set_11	E 0 2 5 7 100 12 15 17 200 22 25 E	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 62,255 74,761 Accepted	Rejected 29,995,611 29,991,030 29,987,580 29,987,580 29,987,580 29,972,183 29,962,290 29,955,775 29,945,350 29,937,745 29,925,239 Rejected	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,937,745 29,925,239 FA	FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,935,745 29,932,745 29,932,745	FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366 25,411 44,377 FA	FR 0 0 0 0 1 0 0 0 0 0 2 1 FR	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067 20,798,178 26,094,659 FA	FR 0
Set_11	E 0 2 5 7 10 12 15 17 20 22 25 E 0	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 62,255 74,761 Accepted 49	Rejected 29,995,611 29,991,030 29,987,580 29,987,580 29,987,580 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,937,745 29,925,239 Rejected 29,999,951	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,937,745 29,925,239 FA 0	FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,945,350 29,937,745 29,925,239 FA 0	FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366 25,411 44,377 FA 53	FR 0 0 0 0 1 0 0 0 0 0 0 2 1 FR 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 9,832,285 16,815,067 20,798,178 26,094,659 FA 0	FR 0
Set_11	E 0 2 5 7 10 12 15 17 20 22 25 E 0 2	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 54,650 54,655 74,761 Accepted 49 163	Rejected 29,995,611 29,991,030 29,987,580 29,987,580 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,925,239 Rejected 29,999,951 29,999,837	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,937,745 29,925,239 FA 0 71	FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,945,350 29,937,745 29,925,239 FA 0 71	FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366 25,411 44,377 FA 53 44	FR 0 0 0 0 1 0 0 0 0 0 0 2 1 FR 0 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067 20,798,178 26,094,659 FA 0 0 55	FR 0
Set_11	E 0 2 5 7 10 12 15 17 20 22 25 6 0 2 5	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 54,650 54,655 74,761 Accepted 49 163 301	Rejected 29,995,611 29,991,030 29,987,580 29,987,580 29,987,580 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,937,745 29,925,239 Rejected 29,999,837 29,999,837 29,999,699	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,937,745 29,925,239 FA 0 71 249	FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	FA 0 1,405 2,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,945,350 29,937,745 29,925,239 FA 0 7 1 0	FR 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366 25,411 44,377 FA 53 44 49	FR 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067 20,798,178 26,094,659 FA 0 55 161	FR 0
Set_11	E 0 2 5 7 10 12 15 17 20 21 22 25 0 22 25 0 2 5 7	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 62,255 74,761 Accepted 49 163 301 375	Rejected 29,995,611 29,991,030 29,987,580 29,987,580 29,984,595 29,972,183 29,962,290 29,955,775 29,945,350 29,945,350 29,925,239 Rejected 29,999,511 29,999,637 29,999,625	FA 0 1,405 12,185 41,555 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,925,239 FA 0 711 249 698	FR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,925,239 69,925,239 60 71 20,925,239 60 71 249 698	FR 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366 25,411 44,377 FA 53 44 49 48	FR 0 0 0 0 0 1 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067 20,798,178 26,094,659 FA 0 555 161	FR 0
12 Set_11	E 0 2 5 7 10 12 15 17 20 21 22 25 6 0 25 7 10 25 7 10 25 7 10 20 21 22 32 33 34 35 37 30	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 62,255 74,761 Accepted 49 163 301 375 472	Rejected 29,995,611 29,991,030 29,987,580 29,987,580 29,984,595 29,972,183 29,962,290 29,955,775 29,945,350 29,945,350 29,925,239 Rejected 29,999,521 29,999,683 29,999,625 29,999,528	FA 0 1,405 12,185 41,555 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,925,239 FA 0 711 249 698 29,995,28	FR 0	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,952,735 29,925,239 69,972,145 29,952,735 29,925,239 60 71 249,995,28	FR 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366 25,411 44,377 FA 53 44 49 48 42	FR 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067 20,798,178 26,094,659 FA 0 555 161 2,122	FR 0
et_12	E 0 2 5 7 100 122 137 200 220 250 6 7 10 112 120 200 210 220 25 7 10 110 120 121	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 62,255 74,761 Accepted 49 163 301 3375 472 520	Rejected 29,995,611 29,991,030 29,987,580 29,984,595 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,945,350 29,925,239 Rejected 29,999,521 29,999,625 29,999,625 29,999,528 29,999,480	FA 0 1,405 12,185 41,555 29,977,286 29,972,183 29,962,290 29,955,775 29,945,350 29,925,239 FA 0 711 249 698 29,999,528 29,999,480	FR 0	FA 0 1,405 12,185 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,937,745 29,937,745 29,945,774 29,937,745 29,947,745 29,947,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,939,742 0 71 29,939,838 29,939,9328 29,939,9480	FR 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366 25,411 44,377 FA 53 44 49 48 42 45	FR 0 0 0 0 0 1 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067 20,798,178 26,094,659 FA 0 0 555 161 2,122 5,627 64,225	FR 0
Set_12 Set_11	E 0 2 5 7 10 12 15 7 20 21 22 25 6 7 10 22 35 7 100 21 35 7 100 110 120 130 140	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 62,255 74,761 Accepted 49 163 301 3375 472 472 520 555	Rejected 29,995,611 29,991,030 29,987,580 29,984,595 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,945,350 29,925,239 Rejected 29,999,521 29,999,625 29,999,625 29,999,425 29,999,425	FA 0 1,405 12,185 41,555 29,977,986 29,962,290 29,965,775 29,945,350 29,925,239 FA 0 711 249 698 29,999,528 29,999,425	FR 0	FA 0 1,405 12,185 29,977,986 29,972,183 29,952,775 29,955,775 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 29,937,745 9,937,745 9,937,745 9,937,745 9,937,745 9,937,745 9,937,745 9,937,745 9,937,745 9,937,745 9,937,745 9,937,745 9,937,745 9,937,745 9,937,945 9,939,9425	FR 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366 25,411 44,377 FA 53 44 49 48 42 45 82	FR 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067 20,798,178 26,094,659 FA 0 555 161 212 5,627 64,225 775,314	FR 0
Set_12 Set_11	E 0 2 5 7 10 12 15 7 20 21 22 25 6 7 10 22 25 7 10 2 5 7 100 12 131 14	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 62,255 74,761 Accepted 49 163 301 3375 472 49 520 555 54,550	Rejected 29,995,611 29,991,030 29,987,580 29,987,580 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,937,745 29,925,239 Rejected 29,999,625 29,999,629 29,999,629 29,999,629 29,999,420 29,999,420 29,999,425 29,999,425 29,999,425 29,999,377	FA 0 1,405 12,185 41,555 29,977,986 29,952,775 29,955,775 29,945,350 29,925,239 FA 0 711 249 698 29,995,288 29,999,528 29,999,425 29,999,377	FR 0	FA 0 1,405 29,977,986 29,972,183 29,927,2183 29,925,230 29,945,350 29,925,239 FA 0 711 249 9 9,952 29,952,828 29,999,428 29,999,528 29,999,425 29,999,377	FR 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366 25,411 44,377 FA 53 44 49 48 42 45 82 175	FR 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067 20,798,178 26,094,659 FA 0 555 161 20,55 161 212 2,5,627 64,225 775,314 2,052,498	FR 0
Set_12 Set_11	E 0 2 5 7 100 12 15 17 20 22 25 6 0 2 5 7 100 2 5 7 100 12 101 12 12 130 141 152 153 154 155 150 151 152 153 154 155 155 155 155 156 157 158 159 150 150 150 150 150 150 150	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 62,255 74,761 Accepted 49 163 301 3375 4472 520 555 623 74,761	Rejected 29,995,611 29,991,030 29,987,580 29,984,595 29,977,986 29,972,183 29,962,290 29,945,350 29,945,350 29,937,745 29,992,239 Rejected 29,999,695 29,999,699 29,999,625 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,377 29,999,327	FA 0 1,405 12,185 41,555 29,977,986 29,952,7183 29,962,290 29,955,775 29,945,350 29,925,239 FA 0 711 249 698 29,999,528 29,999,528 29,999,377 29,999,3282	FR 0	FA 0 1,405 12,185 29,977,986 29,927,183 29,952,757 29,945,350 29,952,757 29,952,757 29,952,757 29,952,757 29,952,757 29,952,757 29,952,757 29,952,82 29,952,82 29,959,528	FR 0	FA 2,933 890 1,704 2,644 4,759 6,729 9,498 12,134 18,366 25,411 44,377 FA 53 44 49 48 42 45 82 175 417	FR 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067 20,798,178 26,094,659 FA 0 555 1611 212 5,627 64,225 775,314 2,052,498 5,679,869	FR 0
Set_12 Set_11	E 0 2 5 7 100 12 15 17 20 22 25 6 0 2 5 7 100 2 5 7 100 12 15 7 100 122 135 140 150 170 122 15 170 122 135 140 150 151 152 153 154 155 155 156 157 158 159 150 150 151 152 <	Accepted 4,389 8,970 12,420 15,405 22,014 27,817 37,710 44,225 54,650 62,255 74,761 Accepted 49 163 301 375 472 520 575 623 7718 842	Rejected 29,995,611 29,991,030 29,987,580 29,987,580 29,987,580 29,972,183 29,962,290 29,955,775 29,937,745 29,925,239 Rejected 29,999,837 29,999,837 29,999,828 29,999,425 29,999,425 29,999,327 29,999,328 29,999,328 29,999,425 29,999,158	FA 0 1,405 12,185 41,555 29,977,986 29,972,183 29,962,290 29,955,775 29,945,350 29,925,239 FA 0 771 249 698 29,995,282 29,999,480 29,999,425 29,999,282 29,999,158	R 0	FA 0 1,405 12,185 29,977,986 29,972,183 29,952,200 29,952,775 29,945,350 29,952,755 29,952,775 29,952,775 29,952,775 29,952,775 29,952,775 29,952,775 29,952,775 29,952,830 29,993,774 29,999,528 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425 29,999,425	FR 0	FA 2,933 890 1,704 2,644 4,759 9,498 12,134 18,366 25,411 44,377 FA 53 44 49 48 42 45 82 175 417 593	FR 0	FA 0 1,173 8,489 24,946 145,053 833,703 5,088,387 9,832,285 16,815,067 20,798,178 26,094,659 FA 0 555 161 2012 5,627 64,225 775,314 2,052,498 5,679,869 10,277,297	FR 0

11 Evaluating the Number of Falsely-Accepted and Falsely-Rejected Pairs Using Single End and Paired End Reads

We assess the accuracy of Shouji using both single end and paired end reads. We first map 3' reads from ERR240727.fastq (i.e., reads from ERR240727_2.fastq) to the human reference genome (GRCh37) using mrFAST (Alkan et al., 2009) with an edit distance threshold of 2. We then use the first 30 million read-reference pairs that are produced by mrFAST before performing alignment to examine the filtering accuracy of Shouji. In Table 12, we show the number of falsely-accepted and falsely-rejected pairs of Shouji using these 30 million pairs over different edit distance thresholds. Generating the read-reference pairs in this way allows us to examine the filtering accuracy of Shouji using both aligned (i.e., pairs that have edits no more than the allowed edit distance threshold) and unaligned (i.e., pairs that have edits more than the allowed edit distance threshold) pairs. We use the same method to generate set 1 from ERR240727 1.fastq, as we describe in Section 3.1 in the main manuscript. We observe that the accuracy of Shouji using 3' reads from ERR240727.fastq remains almost the same as that of Shouji when we use 5' reads from ERR240727.fastq (which we show in Table 9 when we use set 1). Next, we map both 5' reads and 3' reads from ERR240727.fastq to the human reference genome using the mrFAST mapper in paired end mode. We then use the first 30 million read-reference pairs that are produced by mrFAST before performing alignment to examine the filtering accuracy of Shouji. In Table 13, we show the number of falsely-accepted and falsely-rejected pairs of Shouji using these 30 million pairs. We observe the results are similar when using paired end reads as when using single end reads. Based on Table 12 and Table 13, we conclude that the evaluation of our pre-alignment filter does not depend on the paired end sequencing or paired end reads. Similarly with any dynamic programming sequence alignment algorithm, Shouji always examines a single reference segment with a single read individually and independently from the way this pair is generated. The read mapper is responsible for generating the read-reference pairs that must be verified using a dynamic programming sequence alignment algorithm. Shouji examines these pairs (before using the computationally-expensive sequence alignment algorithms) regardless of the algorithm (e.g., single end read mapping or paired end read mapping) used to generate these pairs.

Table 12: Number of falsely-accepted and falsely-rejected sequence pairs of Shouji using single end reads from ERR240727_2.fastq mapped to the human reference genome. We use Edlib (Šošić and Šikić, 2017) to generate the ground truth edit distance value for each sequence pair.

E	Edlib b	aseline	Shouji					
	Aligned	Unaligned	Aligned	Unaligned	Falsely-Accepted	Falsely-Rejected		
0	206,252	29,793,748	206,252	29,793,748	0	0		
1	1,359,165	28,640,835	1,680,722	28,319,278	321,557	0		
2	3,308,445	26,691,555	4,562,146	25,437,854	1,253,701	0		
3	5,673,028	24,326,972	8,290,885	21,709,115	2,617,857	0		
4	7,929,996	22,070,004	12,171,061	17,828,939	4,241,065	0		
5	9,920,919	20,079,081	16,051,171	13,948,829	6,130,252	0		
6	11,710,868	18,289,132	20,532,091	9,467,909	8,821,223	0		
7	13,409,936	16,590,064	23,845,857	6,154,143	10,435,921	0		
8	15,078,030	14,921,970	26,405,117	3,594,883	11,327,087	0		
9	16,727,424	13,272,576	27,901,872	2,098,128	11,174,448	0		
10	18,339,408	11,660,592	28,680,484	1,319,516	10,341,076	0		

Table 13: Number of falsely-accepted and falsely-rejected sequence pairs of Shouji using paired end reads from ERR240727.fastq mapped to the human reference genome. We use Edlib (Šošić and Šikić, 2017) to generate the ground truth edit distance value for each sequence pair.

E	Edlib baseline		Shouji					
E	Aligned	Unaligned	Aligned	Unaligned	Falsely-Accepted	Falsely-Rejected		
0	0	30,000,000	0	30,000,000	0	0		
1	373,921	29,626,079	453,808	29,546,192	79,887	0		
2	1,318,319	28,681,681	1,947,127	28,052,873	628,808	0		
3	3,207,952	26,792,048	5,224,261	24,775,739	2,016,309	0		
4	5,500,950	24,499,050	9,227,434	20,772,566	3,726,484	0		
5	7,709,237	22,290,763	13,305,866	16,694,134	5,596,629	0		
6	9,698,512	20,301,488	18,208,145	11,791,855	8,509,633	0		
7	11,529,693	18,470,307	22,281,600	7,718,400	10,751,907	0		
8	13,293,029	16,706,971	25,736,052	4,263,948	12,443,023	0		
9	15,041,936	14,958,064	27,833,759	2,166,241	12,791,823	0		
10	16,782,466	13,217,534	28,890,050	1,109,950	12,107,584	0		

12 FPGA Acceleration of Shouji and MAGNET

We analyze the benefits of accelerating the CPU implementation of our pre-alignment filters Shouji and MAGNET using FPGA hardware. As we show in Table 14, our hardware accelerators are two to three orders of magnitude faster than the equivalent CPU implementations of Shouji and MAGNET.

Table 14: Execution time (in seconds) of the CPU	implementations of Shouji and MAGNET	filters and that of their	hardware-accelerated
versions (using a single filtering unit).			

Ε	Shouji-CPU Shouji-FPGA		Speedup	MAGNET-CPU	MAGNET-FPGA	Speedup			
Sequence Length = 100									
2	474.27	2.89	164.11x	632.02	2.89	218.69x			
5	1,305.15 2.89 451.61x		1,641.57	2.89	568.02x				
Sequence Length $= 250$									
2	1,689.09	2.89*	584.46x	5,567.62	2.89*	1,926.51x			
5	6,096.61	2.89*	2,109.55x	14,328.28	2.89*	4,957.88x			

* Estimated based on the resource utilization and data throughput

13 Execution time breakdown of Read Mapping combined with Shouji

We provide the total runtime breakdown of mrFAST (v. 2.6.1) (Alkan et al., 2009) and BWA-MEM (Li, 2013) with Shouji as a pre-alignment filter. We break down the execution time of read mapping with Shouji into 1) read-reference pair generation time, 2) Shouji filtering time, 3) Shouji pre-processing time, 4) Shouji transfer time, and 5) dynamic programming alignment time. The sum of these five runtime values provides the total execution time of mrFAST (v. 2.6.1 that includes FastHASH (Xin et al., 2013)) (Alkan et al., 2009) and BWA-MEM (Li, 2013) with Shouji compared to the baseline (i.e., the last column of Table 15 represents the runtime of mrFAST and BWA-MEM without Shouji) in Table 15. We map all reads from ERR240727_1 (100 bp) to GRCh37 with an edit distance threshold of 2% and 5%. Based on Table 15, we make the following key observation: the dynamic programming alignment time drops by a factor of 4-24 (the 7th column of Table 15 compared with the 10th column of Table 15) after integrating Shouji with read mapping as a pre-alignment step.

We conclude that the ability of Shouji to accelerate read mapping scales very well over a wide range of edit distance threshold values.

Table 15: Total execution time breakdown (in seconds) of mrFAST and BWA-MEM with and without Shouji, for an edit distance threshold of 2% and 5%. The green shaded columns represent the processing time spent by each step of the original read mapper (without Shouji). The orange and blue shaded columns represent the processing time spent by each step of the accelerated read mapper (with the addition of Shouji as a pre-alignment step). The orange shaded columns represent the processing time spent by Shouji on the FPGA board and the host CPU.

	E	Read mapping time with Shouji						Read mapping time without Shouji (baseline)		
		Read-ref pair generation time	Shouji (FPGA) . filtering time	Shouji (CPU)		Alignment	Total	Read-ref pair	Alignment	Total
				pre- processing	Transfer time	time	time	time	time	time
mrFAST	2	175.02	0.0616	3.2239	0.2919	16.6929	195.2902	175.02	67.08	242.1
	5	198.02	1.3176	53.9911	6.2457	242.8571	502.4315	198.02	2333.99	2532.01
BWA-MEM	2	622.1	0.0010	0.0516	0.0050	4.8219	626.9794	622.1	46.02	668.12
	2*	623.03	0.0124	0.6477	0.0622	2.0729	625.8252	623.03	47.08	670.11
	5	649.02	0.0010	0.0521	0.0050	4.7089	653.7870	649.02	46.12	695.14
	5*	650.01	0.0129	0.6740	0.0647	1.9190	652.6806	650.01	46.08	696.09

14 Edlib, Parasail, SHD, mrFAST, and BWA-MEM Configurations

In Table 16, we list the software packages that we cover in our performance evaluation, including their version numbers and function calls used.

Table 16: Read aligners and pre-alignment filters used in our performance evaluations.

Edlib: November 5 2017

```
Banded Levenshtein Distance:

EdlibAlignResult resultEdlib = edlibAlign(RefSeq, ReadLength, ReadSeq, ReadLength, edlibNewAlignConfig(ErrorThreshold,

EDLIB_MODE_NW, EDLIB_TASK_PATH, NULL, 0));

edlibFreeAlignResult(resultEdlib);

if (resultEdlib.editDistance!= -1)

Accepted =1;

else Accepted =0;

Banded Levenshtein Distance with backtracking:

EdlibAlignResult resultEdlib = edlibAlign(RefSeq, ReadLength, ReadSeq, ReadLength, edlibNewAlignConfig(ErrorThreshold,

EDLIB_MODE_NW, EDLIB_TASK_PATH, NULL, 0));

char* cigar = edlibAlignmentToCigar(resultEdlib.alignment, resultEdlib.alignmentLength, EDLIB_CIGAR_STANDARD);

free(cigar);

edlibFreeAlignResult(resultEdlib);

Parasail: January 7 2018

function = parasail_lookup_function("nw_banded");
```

runction = parasail_tookup_function(_nw_banded_); result = function(RefSeq, ReadLength, ReadSeq, ReadLength,10, 1, ErrorThreshold,¶sail_blosum62); if(parasail_tracedeck_generic(RefSeq, ReadLength, ReadSeq, ReadLength, "Query:", "Target:", ¶sail_blosum62, result, '|', '.', 50, 14, 0); if (result->score != 0) { cigar2=parasail_result_get_cigar(result, RefSeq, ReadLength, ReadSeq, ReadLength, ¶sail_blosum62); parasail_cigar_free(cigar2); } }

SHD: November 7 2017, compiled using g++-4.9

for (k=1;k<=1+ (ReadLength/128);k++)

totalEdits= totalEdits + (bit_vec_filter_ssel(read_t, ref_t, length, ErrorThreshold));

mrFAST: November 29 2017

./mrfast-2.6.1.0/mrfast --search human_g1k_v37.fasta --seq ../ERR240727_1_100bp.fastq -e 2

The human reference genome can be downloaded from: <u>ftp://ftp.ncbi.nlm.nih.gov/1000genomes/ftp/technical/reference/human_g1k_v37.fasta.gz</u>

Extracting read-reference pairs:

- 1- Add the following to line 1786 of https://github.com/BilkentCompGen/mrfast/blob/master/MrFAST.c
- 2- Extract reference segment:
- for (n = 0; n < 100; n++) printf("%d", _msf_refGen[n + genLoc + _msf_refGenOffset 1 leftSeqLength]);
- 3- Extract read sequence: printf("\t%s\n", _tmpSeq);

BWA-MEM: November 25 2018

/bwa mem -w 3 ../human_g1k_v37.fasta ../../../Desktop/Filters_29_11_2016/ERR240727_1_100bp.fastq

Report all secondary alignments: /bwa mem -a -w 3 ./human g1k_v37.fasta ./././Desktop/Filters_29_11_2016/ERR240727_1_100bp.fastq

Extracting read-reference pairs:

- 1- Add the following code between line 166 and line 167 of https://github.com/lh3/bwa/blob/master/bwa.c
- 2- Extract reference segment:
- for (i = 0; i < rlen; ++i) putchar("ACGTN"[(int)rseq[i]]); putchar('\t');
- 3- Extract read sequence: for (i = 0; i < l_query; ++i) putchar("ACGTN"[(int)query[i]]); putchar('\n');</p>

REFERENCES

- Alkan, C., Kidd, J. M., Marques-Bonet, T., Aksay, G., Antonacci, F., Hormozdiari, F., Kitzman, J. O., Baker, C., Malig, M. and Mutlu, O. (2009) Personalized copy number and segmental duplication maps using next-generation sequencing, *Nature genetics*, **41**, 1061-1067.
- Alser, M., Hassan, H., Xin, H., Ergin, O., Mutlu, O. and Alkan, C. (2017) GateKeeper: a new hardware architecture for accelerating pre-alignment in DNA short read mapping, *Bioinformatics*, **33**, 3355-3363.
- Alser, M., Mutlu, O. and Alkan, C. (July 2017) Magnet: Understanding and improving the accuracy of genome pre-alignment filtering, *Transactions on Internet Research* 13.
- Li, H. (2013) Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM, arXiv preprint arXiv:1303.3997.
- McNamara, M. (2001) IEEE Standard Verilog Hardware Description Language. The Institute of Electrical and Electronics Engineers, *Inc. IEEE Std*, 1364-2001.
- Šošić, M. and Šikić, M. (2017) Edlib: a C/C++ library for fast, exact sequence alignment using edit distance, *Bioinformatics*, **33**, 1394-1395.
- Xilinx (November 17, 2014) 7 Series FPGAs Configurable Logic Block User Guide. Xilinx.
- Xin, H., Greth, J., Emmons, J., Pekhimenko, G., Kingsford, C., Alkan, C. and Mutlu, O. (2015) Shifted Hamming Distance: A Fast and Accurate SIMD-Friendly Filter to Accelerate Alignment Verification in Read Mapping, *Bioinformatics*, **31**, 1553-1560.
- Xin, H., Lee, D., Hormozdiari, F., Yedkar, S., Mutlu, O. and Alkan, C. (2013) Accelerating read mapping with FastHASH, BMC genomics, 14, S13.