Note S1 Methods

The schema of the proposed method is illustrated in Fig. S1. Smash++ takes a reference and a target file, as inputs, and produces a position file, as output, which is then fed to the Smash++ visualizer to produce an SVG image. This process has eight major stages: (1) compression of the original target file based on the model of original reference file, (2) filtering and segmentation of the compressed file, (3) reference-free compression of the segmented files that are obtained by the previous stage, (4) compression of the original reference file based on the model of segmented files obtained by stage 2, (5) filtering and segmentation of the compressed files, (6) reference-free compression of the segmented files which are obtained by the stage 5, (7) aggregating positions based on the results of stages 3 and 6, and (8) visualizing the positions. The following sections describe the process in detail.

Fig. S1. The schema of Smash++.

S1.1 Data modeling

Smash++ works on the basis of cooperation between finite-context models (FCMs) and substitutional tolerant Markov models (STMMs), as illustrated in Fig. S2. Applying these models on various contexts, seen in the past, provides probability and weight values, which are then mixed to provide the final probability (P) of the next symbol, G in this case. The following subsections describe FCMs and STMMs in detail.

Fig. S2. model.

Finite-context model (FCM) A finite-context model considers Markov property to estimate the probability of the next symbol in an information source, based on the past k symbols (a context of size k) [1–3]. Denoting the context as $x_{i-k}^{i-1} = x_{i-k}x_{i-k+1}\cdots x_{i-2}x_{i-1}$, the probability of the next symbol s, which is posed at i, can be estimated as

$$P_m(s|x_{i-k}^{i-1}) = \frac{N(s|x_{i-k}^{i-1}) + \alpha}{N(x_{i-k}^{i-1}) + \alpha|\Theta|}, \qquad \textbf{(Eq. S1)}$$

in which m stands for model (FCM in this case), $N(s|x_{i-k}^{i-1})$ shows the number of times that the information source has generated symbol s in the past, $|\Theta|$ denotes size of the alphabet Θ, $N(x_{i-k}^{i-1}) = \sum_{j\in\Theta} N(j|x_{i-k}^{i-1})$ represents the total number of events occurred for the context x_{i-k}^{i-1} and α allows to keep a balance between the maximum likelihood estimator and the uniform distribution. Eq. S1 turns to the Laplace estimator, for $\alpha = 1$, and also behaves as a maximum likelihood estimator, for large number of events i [4].

Substitutional tolerant Markov models A substitutional tolerant Markov model (STMM) [5,6] is a probabilistic-algorithmic context model. An STMM uses the occurrence probabilities stored in the memory and assumes that the symbol to be seen in the sequence is the one with the highest probability. This way, it does not take into consideration the symbol that actually is in the sequence.

Considering the k most recent symbols, the probability of the next symbol s is estimated as

$$P_m(s|{x'}_{i-k}^{i-1}) = \frac{N(s|{x'}_{i-k}^{i-1}) + \alpha}{N({x'}_{i-k}^{i-1}) + \alpha|\Theta|}, \qquad \textbf{(Eq. S2)}$$

in which N is the memory counts regarding the model and x' is a copy of the context x that is modified as

$$x'_i = \underset{\forall s \in \Theta}{\operatorname{argmax}} P_m(s|x'^{i-1}_{i-k}). \qquad \textbf{(Eq. S3)}$$

An STMM is an algorithmic model, namely because it can be disabled and enabled based on its performance. This operation is done considering a predefined threshold t. For this purpose, a cache array (history) with the size of k (context-order size) is used to store the recent k hits/misses. When we see a symbol in the sequence, if it is the most probable symbol, based on the number of occurrences saved in the memory, a hit will be stored in the history array; otherwise, a miss will be stored. At the time of seeing a symbol, before storing any hit/miss in the history, we check the number of misses. If it is greater than the threshold t, the STMM will be disabled and the history will be reset. This process starts over again for the next symbol.

To make the difference of FCM and STMM clearer, we give an example. Assume, the current context is c_0 = FDCAE, with $k = 5$, and the number of occurrences stored in the memory are F = 8, D = 5, C = 12, A = 7 and E = 10. Also, assume the next symbol in the sequence, which is to be compressed, is A. Based on an FCM, the next context will be c_1 = DCAEA, while based on an STMM, it will be c'_1 = DCAEC. This is because an FCM considers the next symbol as the actual one seen in the sequence, i.e. A, but an STMM considers it as the most probable symbol, C. This way, the STMM assumes the next symbol to be compressed is C, instead of A.

Cooperation of FCMs and STMMs In the case of cooperation between FCMs and STMMs, considering the k most recent symbols in the sequence, the probability of the next symbol s is estimated as

$$P(s) = \sum_{m \in \mathcal{F}} P_m(s|x^{i-1}_{i-k})\, w_{m,i} + \sum_{m \in \mathcal{S}} P_m(s|x'^{i-1}_{i-k})\, w_{m,i}, \qquad \textbf{(Eq. S4)}$$

where $\mathcal{F}$ is the set of FCMs and $\mathcal{S}$ is the set of STMMs. $P_m(s|x^{i-1}_{i-k})$ and $P_m(s|x'^{i-1}_{i-k})$ are the probabilities of the next symbol estimated by the FCM and the STMM, respectively. Also, $w_{m,i}$ is a weight assigned to each model, based on its performance. For this weighting factor, we have

$$\begin{aligned} \forall m \in \mathcal{F}: \quad & w_{m,i} \propto (w_{m,i-1})^{\gamma_m} P_m(s|x^{i-2}_{i-k-1}) \\ \forall m \in \mathcal{S}: \quad & w_{m,i} \propto (w_{m,i-1})^{\gamma_m} P_m(s|x'^{i-2}_{i-k-1}), \end{aligned} \qquad \textbf{(Eq. S5)}$$

in which $\gamma_m \in [0, 1)$ acts as a forgetting factor for the models. Note,

$$\sum_{m \in \mathcal{F}} w_{m,i} + \sum_{m \in \mathcal{S}} w_{m,i} = 1. \qquad \textbf{(Eq. S6)}$$

We have found that the lower the context-order size k, the lower γ_m should be assigned to the model, and vice versa. For example, for a model with $k = 6$, a $\gamma_m \simeq 0.9$ and for a model with $k = 18$, a $\gamma_m \simeq 0.95$ is the appropriate choice. This means, the more complex the model, the less the forgetting intensity.

S1.2 Finding similar regions

In order to smooth the profile information, we use Hann window [7], which is a discrete window function given by

$$w[n] = 0.5 - 0.5\ \cos\left(\frac{2\pi n}{N}\right) = \sin^2\left(\frac{\pi n}{N}\right), \qquad \textbf{(Eq. S7)}$$

in which, $0 \le n \le N$ and length of the window is $N + 1$ (Fig. S3).

Fig. S3. Hann window for 101 samples.

S1.3 Computing complexities

S1.4 The software

Besides Hann window, that is used as default to filter the profile information obtained by the reference-based compression, we have implemented several other window functions (Fig. S4), including Blackman [7], Hamming [8], Nuttall [9], rectangular [10], sine [11], triangular [12] and Welch [13] windows. These functions are given by

$$\begin{aligned}
&w[n] = 1, && \text{(rectangular)}\\
&w[n] = 1 - \left|\tfrac{n-N/2}{L/2}\right|, \quad L = N, && \text{(triangular/Bartlett)}\\
&w[n] = 1 - \left(\tfrac{n-N/2}{N/2}\right)^2, && \text{(Welch)}\\
&w[n] = \sin\left(\tfrac{\pi n}{N}\right), && \text{(sine)}\\
&w[n] = 0.54348 - 0.45652\ \cos\left(\tfrac{2\pi n}{N}\right), && \text{(Hamming)}\\
&w[n] = 0.42659 - 0.49656\ \cos\left(\tfrac{2\pi n}{N}\right) + 0.07685\ \cos\left(\tfrac{4\pi n}{N}\right), && \text{(Blackman)}\\
&w[n] = 0.35577 - 0.48740\ \cos\left(\tfrac{2\pi n}{N}\right) + 0.14423\ \cos\left(\tfrac{4\pi n}{N}\right) - 0.01260\ \cos\left(\tfrac{6\pi n}{N}\right). && \text{(Nuttall)}
\end{aligned}$$

(Eq. S8)

Fig. S4. Window functions.

References

[1] K. Sayood, *Introduction to data compression.* Morgan Kaufmann, 2017.

[2] M. Hosseini, D. Pratas, and A. J. Pinho, "AC: A compression tool for amino acid sequences," *Interdisciplinary Sciences: Computational Life Sciences*, vol. 11, no. 1, pp. 68–76, 2019.

[3] A. J. Pinho and D. Pratas, "MFCompress: a compression tool for FASTA and multi-FASTA data," *Bioinformatics*, vol. 30, no. 1, pp. 117–118, 2013.

[4] D. Pratas, R. M. Silva, A. J. Pinho, and P. J. Ferreira, "An alignment-free method to find and visualise rearrangements between pairs of DNA sequences," *Scientific reports*, vol. 5, p. 10203, 2015.

[5] D. Pratas, A. J. Pinho, and P. J. Ferreira, "Efficient compression of genomic sequences," in *Proceedings of DCC '16: Data Compression Conference.* IEEE, 2016, pp. 231–240.

[6] D. Pratas, M. Hosseini, and A. J. Pinho, "Substitutional tolerant Markov models for relative compression of DNA sequences," in *International Conference on Practical Applications of Computational Biology & Bioinformatics (PACBB).* Springer, 2017, pp. 265–272.

[7] R. Blackman and J. Tukey, "Particular pairs of windows," *The measurement of power spectra, from the point of view of communications engineering*, pp. 95–101, 1959.

[8] J. W. Tukey and R. W. Hamming, *Measuring noise color.* Bell Telephone Laboratories, 1949.

[9] A. Nuttall, "Some windows with very good sidelobe behavior," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 1, pp. 84–91, 1981.

[10] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing.* Upper Saddle River, NJ: Prentice Hall, 1999.

[11] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[12] M. S. Bartlett, "Periodogram analysis and continuous spectra," *Biometrika*, vol. 37, no. 1/2, pp. 1–16, 1950.

[13] P. Welch, "The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms," *IEEE Transactions on audio and electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967.