Picking an Imperfect Palindrome

Georgii A. Khaziev, Oleg A. Zverkov and Alexandr V. Seliverstov

Abstract. Automatic search of sequences, that are close to perfect palindromes remains an open problem. We discuss an algorithm `de_shapker` which decreases the value of imp function by cutting non-complement subsequence. Described algorithm allows picking non-palindromic sequences, that are close to palindromes.

Introduction

In computational biology nucleotide sequence is defined as sequence over four-letter alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. The involution $\mathrm{c} : \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^n \to \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^n$ is called reverse complement and defined as follows

$$\begin{cases} \mathrm{c}(\mathtt{A}) = \mathtt{T} \\ \mathrm{c}(\mathtt{T}) = \mathtt{A} \\ \mathrm{c}(\mathtt{G}) = \mathtt{C} \\ \mathrm{c}(\mathtt{C}) = \mathtt{G} \\ \mathrm{c}(xy) = \mathrm{c}(y)\mathrm{c}(x) \end{cases} \tag{1}$$

A sequence x is called perfect palindrome if $x = \mathrm{c}(x)$. A sequence is called imperfect palindrome otherwise. But automatic picking of imperfect palindromes, that are close to perfect ones remains an open problem. This problem is close to repeats detection problem, described at [1, 2].

In previous works we proposed an algorithm `palindrome_self_alignment`, that gets sequence x as an input and finds such partition $x = wz$ that edit distance between x and $w\mathrm{c}(w)$ is minimal [3]. Also, we introduced function

$$\mathrm{imp}(x) = \frac{\min\{\mathrm{dist}(x, w\mathrm{c}(w)) | x = wz\}}{|x|}. \tag{2}$$

The closer value of $\mathrm{imp}(x)$ to 0, the closer x to being a perfect palindrome. But nucleotide sequences that can be found in real biological data can be far from

perfect palindrome by itself, but have long subsequences, that are close to perfect palindromes. To allocate such subsequences we introduced algorithms called trimmers [4]. The main idea behind those algorithms is finding potential non-complement parts – termini, that are located on the ends of the sequence. So even if $\mathrm{imp}(x)$ value is greater than given threshold, after which x is considered close to palindrome, after trimming a substring of x could be found with the value lower, than threshold.

1. Deleting loop

After trimming of x, resulted sequence can still contain a non-complement substring inside, negatively affecting value of imp function. This substring is called loop, and plays a vital role in formation of secondary structure of sequence x. To delete this substring we propose the `de_shapker` algorithm. It takes as an input a string x, a floating point value `imp_given`, which equals $\mathrm{imp}(x)$, matrix `H` of values of function $h(j,k)$ from `palindrome_self_alignment`(x) algorithm, integers `iteration_counter`, `window_delta` and `window_size`, and a Python function `strategy`.

Let $x_{result} = x$. On each iteration `de_shapker` algorithm computes l_1 norms of all continuous submatrices $S \in \mathbb{R}^{\texttt{window\_size}\times\texttt{window\_size}}$ of H defined as

$$l_1(S) = \sum_{i=1}^{\texttt{window\_size}} \sum_{i=j}^{\texttt{window\_size}} (S)_{ij} \tag{3}$$

After that, the submatrix $S_{\min}$ with minimal norm is chosen. Then, the index i of the beginning of potential loop in x is computed as $i = \texttt{strategy}(u, v)$, where u and v are coordinates of top-left element of $S_{\min}$ in H. We recommend to choose as a `strategy` a function biased towards minimal value between u and v (e.g. $\min(u, v)$ or $\lfloor \mathrm{mean(u, v)} \rfloor$) as they tend to capture loops more precisely. Subsequently, x_b is computed as x with cutted symbols from i to $i + \texttt{window\_size}$ positions. If $\mathrm{imp}(x_b) < \texttt{imp\_given}$, then x_{result} will be overwritten with x_b, and `imp_given` will be overwritten with $\mathrm{imp}(x_b)$. Next, the `window_size` is increased by $\texttt{window}_\delta$ and the next iteration starts. If $\mathrm{imp}(x_b) \geq \texttt{imp\_given}$, then algorithm returns x_{result}.After all iterations, algorithm returns x_{result}.

To minimize computation time, every l_1 norm on new iteration can be computed from l_1 norms from previous iteration by adding necessary rows and columns.

Theorem 1.

$$\mathrm{imp}(\texttt{de\_shapker}(x, \dots)) \leq \mathrm{imp}(x) \tag{4}$$

Proof. This is true, because on each iteration algorithm only saves new substring of x if imp is decreased. Otherwise, `de_shapker` returns last substring with decreased imp value. □

Conclusion

Discussed algorithm could be useful for picking sequences, that are not close to perfect palindromes by themself, but have long subsequences that are.

2. Funding

The research was carried out within the state assignment of Ministry of Science and Higher Education of the Russian Federation for IITP RAS.

References

[1] Zhao X.X., Wang Z.X., Tang D. et al. *The expectation and the variance of the weights of de Bruijn sequences*, Des. Codes Cryptogr. 2025. https://doi.org/10.1007/s10623-025-01729-2

[2] Nazipova N.N. *Application of suffix arrays to detect repeats in genomic sequences*, Mathematical Biology and Bioinformatics, 2025. V. 20 I. 2. P. 348-362. https://doi.org/10.17537/2025.20.348

[3] Zverkov O., Seliverstov A., Shilovsky G. *Alignment of a Hidden Palindrome*, Mathematical biology and bioinformatics. 2024. V.19. I.2. P. 427-438. https://doi.org/10.17537/2024.19.427

[4] Khaziev G.A., Seliverstov A.V., Zverkov O.A. *Searching for an imperfect palindrome*, Computer algebra: 6th International Conference Materials, Moscow, Russia, June 23–25 2025, RUDN University, 2025, P. 62–65.

Georgii A. Khaziev
Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute)
Moscow, Russia
e-mail: khaziev@iitp.ru

Oleg A. Zverkov
Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute)
Moscow, Russia
e-mail: zverkov@iitp.ru

Alexandr V. Seliverstov
Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute)
Moscow, Russia
e-mail: slvstv@iitp.ru