

Finding Approximate Tandem Repeats with the Burrows-Wheeler Transform

Agnieszka Danek, Rafał Pokrzywa

Abstract—Approximate tandem repeats in a genomic sequence are two or more contiguous, similar copies of a pattern of nucleotides. They are used in DNA mapping, studying molecular evolution mechanisms, forensic analysis and research in diagnosis of inherited diseases. All their functions are still investigated and not well defined, but increasing biological databases together with tools for identification of these repeats may lead to discovery of their specific role or correlation with particular features. This paper presents a new approach for finding approximate tandem repeats in a given sequence, where the similarity between consecutive repeats is measured using the Hamming distance. It is an enhancement of a method for finding exact tandem repeats in DNA sequences based on the Burrows-Wheeler transform.

Keywords—approximate tandem repeats, Burrows-Wheeler transform, Hamming distance, suffix array

I. INTRODUCTION

TANDEM repeats (TRs), which are consecutive, repeating patterns in genomic sequences, belong to the most important loci in genomes, both in the aspect of evolutionary mechanisms of their development and their functional roles in cellular mechanisms. Evolution of tandem repeats loci is governed by a special mechanism, called slippage mutation e.g., [1], which due to its high intensity belongs to major factors of genomic dynamics. A very important issue is dynamics of interaction between slippage and point mutation [2]. As functional elements of genomic sequences, TRs can play important roles in the gene expression and transcription regulations [3]. Some, when occurring in increased, abnormal number, are known to be the cause of inherited diseases (i.e. trinucleotide repeat disorders) [4]. Due to their high abundance in DNA TRs are used as markers for DNA mapping and fingerprinting [5].

Functionalities of TRs in genomic sequences are still not well defined and understood. However, growing biological databases together with tools for efficient and accurate identification of these repeats may lead to discovery of their specific role or correlation with particular diseases or features. In order to develop knowledge on evolutionary mechanisms and functions of TRs as well as to develop methodologies of applications of TRs as genomic markers, efficient tools for their detection are necessary. Detection tools for TRs can be divided into two classes. The first class includes experimental techniques [6] based on DNA amplification methodologies. The second class of methods involves text searches through genomic databases. TR loci discovered by using procedures for

text searches can be potentially used for developing laboratory procedures for disease, forensics or fingerprinting diagnostics.

This paper is devoted to methodologies of detection of TRs by searching through genomic databases. In the aspect of approaches to solutions of this problem, TRs are divided into two groups, exact tandem repeats (ETRs) and approximate tandem repeats (ATRs). For both groups of TRs many detection methods were published, some are overviewed in the recent survey [7]. However, as observed in [7], detection methods for TRs still need development and refinement due to their limitations and differences seen between various approaches. This observation is particularly important for ATRs, due to the fact that different approaches not only differ in algorithmic aspects but also often use different measures of similarity between TR motifs, which makes the result more difficult to compare.

Here a novel algorithm for an exhaustive search of ATRs with substitution mutations is presented. The presented algorithm is a development of an already functional tool for searching for exact tandem repeats [8], [9] based on the use of the Burrows-Wheeler Transform [10] on a searched string. The proposed methodology makes use of an alphabetically sorted array of all input string suffixes, without the need of storing the whole structure in the memory. It can be achieved by applying the Burrows-Wheeler Transform (BWT) to the input string and utilizing the Ferragina and Manzini idea [11] together with two auxiliary arrays of length equal to the input string length. We present our methodology, examples of its application and comparisons with existing tools for detection of ATRs.

The paper is organized as follows. Section II-A define the problem to be solved. A description of the algorithm is presented in section II-B and in section II-C some modifications and improvements are introduced. Section III shows results of some experiments with the implemented algorithm together with comparisons to other algorithms for ATRs discovery and concludes the paper.

II. METHODS

The Burrows-Wheeler Transform is a block-sorting algorithm which permutes the input text according to the sort procedure [10]. First, by applying a shift rotation to the input string, an array of all suffixes of the original string (together with an additional terminal element) is created. Next, all these suffixes are ordered lexicographically. The last column of the created, alphabetically ordered array of suffixes of the input string is a Burrows-Wheeler transformed input string. Such

A. Danek, R. Pokrzywa are with the Institute of Informatics, Silesian University of Technology, Gliwice, Poland (e-mail: agnieszka.danek@polsl.pl, rafal.pokrzywa@polsl.pl).

a rearrangement tends to group the same elements together and thus make the transformed text much more amenable to compression. The transformation is reversible, thus BWT can be employed in a lossless data compression algorithm. The technique of transformation is also convenient in string searching algorithms. A Burrows-Wheeler transformed string, together with structural properties of the suffix array form kind of compressed suffix array. The algorithm presented by Ferragina and Manzini [11] makes use of this structure, to perform a fast, backward search in the string.

An algorithm presented here transforms the input string with BWT [10] and makes use of backward search idea by Ferragina and Manzini [11] to look for ATRs within the input string. The algorithm is an enhancement of the existing one, looking for the ETR and based on BWT. However, in case of searching for ATR the problem becomes more complicated, starting with appropriate definition of TRs that should be found. Below, our algorithm is presented in detail.

A. Definitions

Let us first introduce some terminology connected with exact tandem repeats. *Double exact tandem repeat* of length $2p$ is a string consisting of two consecutive exact copies of a pattern of length p . The length p of the repeated pattern is called a *period* of the tandem repeat. *Multiple exact tandem repeat with period p* (or simply *exact tandem repeat*) is any string of length $n \geq 2p$ for which every substring of length $2p$ is a *double exact tandem repeat*. The ratio $\frac{n}{p}$ is called an *exponent* of the *tandem repeat*. Finally, *maximal exact tandem repeat* is a string which is an *exact tandem repeat* and cannot be further extended to left or right to still meet the definition of the *exact tandem repeat*.

As we are interested in the situation when consecutive copies are inexact (due to mutation), we need to modify the above definitions to the approximate case and introduce a measure of dissimilarity of two strings. Given two strings A and B of equal length, let $h(A, B)$ be the Hamming distance between A and B , that is a minimal number of substitution needed to be done in string A to transform it to string B . *Double K -mismatch tandem repeat* of length $2p$ is a string consisting of two consecutive strings S_1 and S_2 , both of length p , that $h(S_1, S_2) \leq K$. *Multiple K -mismatch tandem repeat with period p* (or simply *K -mismatch tandem repeat*) is any string of length $n \geq 2p$ for which every substring of length $2p$ is a *double K -mismatch tandem repeat*. As in the exact case, the ratio $\frac{n}{p}$ is called an *exponent* of the *tandem repeat*. Finally, *maximal K -mismatch tandem repeat* (from now on referred to as *maximal K -ATR*) is a string which is a *K -mismatch tandem repeat* and cannot be further extended to left or right to still meet the definition of the *K -mismatch tandem repeat*.

Assuming we are given a string S over the alphabet Σ and some maximal number of errors K that are acceptable, we are interested in finding all *maximal K -ATR*s within the string S .

B. Algorithm - The basic idea

The basic algorithm described in this section takes as an input a string S over the alphabet Σ and maximum number of mismatches K . It finds all *maximal K -ATR*s in the string S .

The algorithm can be divided into three steps. In the first step candidates for *double k -mismatch tandem repeat* (where $k \leq K$) are found. The second stage consists of investigating found candidates and accepting or rejecting them. Finally, in the last stage, located *double k -mismatch repeats* are extended to be maximal. Detailed description of each stage is provided below.

1) Finding candidates

The goal of the first stage is to find all pairs of repeats in the input string S that lay appropriately close to each other to be considered a candidate for a *double k -mismatch tandem repeats* (where $k \leq K$).

Initially, to find all groups of repeats occurring in the input string (regardless of their positions), idea similar to the one from the algorithm for finding exact tandem repeats with application of BWT [8], [9] is utilized. The input string is converted according to the BWT [10]. An array Pos with length equal to the input string, determining the original position of the suffixes and an auxiliary array Prm , being an inverse of Pos ($Prm(Pos(i)) = i$), are calculated. Converted input string, together with these arrays, allow to make use of the alphabetically sorted array of input string suffixes, without the need of storing the whole suffix array structure. The algorithm finds the range of positions in the suffix array of all patterns repeated in the string S . It starts with an empty pattern P and recursively appends in front of P each character c from the considered alphabet. This approach uses the results from the previous iteration to calculate a range of positions for a longer pattern ($c + P$) and it is done in a constant time, according to the idea of Ferragina and Manzini [11]. If there is only one occurrence of some pattern P , the recursion is stopped. This way it is possible to go recursively through all groups of repeats found in the input string.

To define how far away from each other a certain pair of repeats should be positioned to be considered a candidate for a *double k -mismatch tandem repeat*, let us state an observation, previously made in similar form by Kurtz et al. [12].

Observation *Two strings of length p with the Hamming distance k between them have always a common, matching substring at corresponding positions of length d , such that:*

$$d \geq \left\lfloor \frac{p}{k+1} \right\rfloor$$

Thus, having a group of repeats of length d , we will use them only to find candidates for *double k -mismatch tandem repeats* with period equal to p and the number of mismatches equal exactly to some k (for each k such that $0 \leq k \leq K$) that satisfy the equation from the Observation 1. Although they may be a part of some other *double K -mismatch tandem repeats* with different period or number of mismatches, this restriction reduces redundancies in calculations, as those other repeats will be found anyway, with different group of repeats consid-

ered or different k . A pair of repeats from the considered group of repeats, with indexes i and j in the suffix array, can be a part of a *double k-mismatch tandem repeat* with period p if a difference between their position in the input string is equal to p , that is $Pos(i) - Pos(j) = p$. To find all such pairs of repeats it is enough to check for each repeat with index i from the group, if p positions to left, another repeat from the same group of repeats exists, that is if $Prm(Pos(i) - p)$ is in the range of indexes of the current group of repeats. If so, the pair of repeats of length d is reported as a candidate for a *double k-mismatch tandem repeat* with period p and number of mismatches k .

2) Validating candidates

In this stage it is checked if the reported pair of repeats (each repeat is of length d) is indeed a part of one (or more) *double k-mismatch tandem repeat* with sought period p and exactly k mismatches. Assuming that the index of the left repeat is i , a *double k-mismatch tandem repeat* of length $2p$ can begin anywhere in the input string between positions $Pos(i) - (p-d)$ and $Pos(i)$. The Hamming distance between consecutive strings of length p beginning from all possible positions is measured. If for any of these positions it is equal exactly to k , the *double k-mismatch tandem repeat* is reported at this position. If the Hamming distance is different than k , candidate from the current position is rejected.

3) Extending found double tandem repeats

In the last stage, the *double k-mismatch tandem repeat* found is extended to the left and to the right to obtain a *maximal K-ATR*. Generally, it is done one character at a time. It is simply checked, if a string of the length $2p$ that begins one character to the left from the found *double k-mismatch tandem repeat* is a *maximal K-ATR* (here number of mismatches does not have to be exactly k , it just needs to be less or equal to K). It is enough to remember the exact number of mismatches in the current repeat and compare the two characters, leaving and entering the next double repeat, with the characters at corresponding positions. With these two comparisons, it is possible to calculate number of mismatches in the new double repeat. Then the operation is repeated, going one position to the left each time, as long as the measured Hamming distance is acceptable. The whole procedure is done also for the right edge of the *double k-mismatch tandem repeat* found.

C. Algorithm - Improvements and modifications

1) Control the types or repeats searched

To manipulate what types of repeats should be searched, several input parameters have been introduced, in addition to the maximum number of mismatches K . First is the maximum percentage of error $Kprc$, which defines the maximum ratio of number of mismatches over *period* in every *double tandem repeat*. It gives the opportunity to control the possible error in the repeat in two different ways. Always the more restricting

parameter is taken into account. For example, if $K = 3$ and $Kprc = 0.5$ then for *double tandem repeat* with period 5, the maximum number of mismatches is 2 (it has to be less than or equal to 50% of 5), while for *double tandem repeat* with larger period, the maximum number of mismatches is 3. Other parameters used to restrict kind of repeats searched are: $minPrd$ and $maxPrd$ (defining minimum and maximum period length), $minExp$ (defining minimum exponent value) and $minLength$ (defining minimum length of the found *maximal K-ATRs*). What is important, the algorithm does not filter out the output, to provide only requested types of repeats, but adjust its operation to the input parameters.

2) Trim erroneous edges

Algorithm search for all ATRs that meet the definition of a *maximal K-ATRs*. This means that often the mismatches are present at the edges of found ATRs, just to obtain as *maximal* repeat as possible. However, erroneous edges do not introduce any valuable information about the repeat. Thus, these artificial edges are cut off from the final ATR, as long as it does not decrease its *exponent* below 2.

3) Mark regions with found repeats

Algorithm described so far will find the same *maximal K-ATR* many times, extending different *double k-mismatch tandem repeats* found within that *maximal K-ATR*. To avoid this redundancy, additional array of bits of length equal to the length of the input string S is used. Initially all bits are set to 0. After a *maximal K-ATR* is found, all bits corresponding to its complete position are set to 1. This array of bits is a simple way to mark regions with found *maximal K-ATRs*. That knowledge is used in the stages of finding and validating candidates for *double k-mismatch tandem repeats* as a first verification step. If a candidate is entirely placed within a region already occupied by other, previously found, *maximal K-ATRs*, the candidate is immediately rejected. In other words it is checked, if any bit corresponding to possible position of the *double k-mismatch tandem repeats* is equal to 0. If so, the candidate can (or certainly will in case of validation phase) occupy a new region, not covered yet by any other repeats. This way it will not be redundant with any previously found *maximal K-ATR*. Described change meaningfully decreases amount of computations performed, but also cause that some *maximal K-ATRs* placed entirely within other, longer *maximal K-ATRs* may not be reported (if the larger repeat is found first). However, we do not consider it as a drawback, as all regions with *maximal K-ATRs* will be outputted anyway. Exact positions and character of shorter repeats may be found by playing with input parameters (i.e. by decreasing the maximum period length).

4) Filter out overlapping repeats

The aforementioned marking can lead to an undefined output, that is, repeats contained entirely in larger ones may or may not be outputted (depending on which repeat was found first). To standardize the expected

output, all overlapping repeats (contained entirely in others) are filter out, so the output will never present them. Applying this filtering without marking will lead to output all *maximal K-ATRs* from the input string, each just one time. It would reduce redundancies in the output, but would not decrease the time of computation.

III. RESULTS AND CONCLUSIONS

The algorithm described was implemented in the Java programming language. As results of our paper we present an example outcome of the program, performance estimation based on the program's running time for varying sequences and a quantitative comparison of detected lists of ATRs between two existing ATR searchers and *Our* algorithm.

As in [8] the program was applied to look for ATRs in the Homo sapiens clone C05/443G Complement Receptor 1-Like (CR1L) gene sequence (DQ007071). Results obtained with input parameters: $K = 2$, $Kprc = 20$, $minPrd = 3$, $maxPrd = 100$, $minExp = 2$ and $minLen = 3$ are in Table I. Successive columns determine start and end position of the found ATR within the input string, its period, exponent and finally, complete sequence of the ATR in the last column. Discovered repeats satisfy the demands defined by the input parameters. To estimate the performance, the program was run for five different sequences downloaded from the GenBank database, varying in the total length. Their accession numbers and total lengths are in Table II, together with experiment results that is the number of ATRs found and the total program's running time (average from 10 runs). Parameters applied are: $K = 4$, $Kprc = 40$, $minPrd = 1$, $maxPrd = 100$, $minExp = 2$ and $minLen = 3$. The program was run on Apple MacBook with 2.26 GHz Intel Core 2 Duo and 2GB RAM, running OS X 10.6.8. The results show that despite the running time increases for longer input string, it is still viable to investigate larger sequences.

There are many heuristic, statistically-based methods for finding (ATRs), some methods with a priori specified size of repeated unit and few exhaustive search methods. Tools for finding ATRs differ not only in an applied approach, but, first of all, in a model of searched ATRs. Varying definitions of ATR makes it difficult to compare operation of individual methods, as in fact they may look for slightly different substrings of the input string. Some tools searching for ATRs, available on the Internet:

- mreps [13]
<http://bioinfo.lifl.fr/mreps/mreps.php>
- Tandem Repeat Finder (TRF) [14]
<http://tandem.bu.edu/trf/trf.html>
- ATRHunter [15]
<http://bioinfo.cs.technion.ac.il/atrhunter>
- Tandem Repeats Database (TRedD) [16]
<http://tandem.sci.brooklyn.cuny.edu/>
- TandemSWAN: Tandem Structure World Analyzer [17]
<http://favorov.imb.ac.ru/swan>
- A Program for Approximate Tandem Repeat [18]
<http://csweb.haifa.ac.il/library/approProg.htmlProgram.do>

The operation of *Our* program was compared with two other tools: Tandem Repeat Finder (TRF) [14] (version 4.04) and mreps [13] (version 2.5), choosing parameters as similar as possible to the ones in *Our* algorithm. As for mreps, that mean that its parameter *resolution* is equal to K (it is not possible to set percentage of mismatch between adjacent copies and *resolution* is close in meaning to K) and small sequences are allowed (*allowsmall* is set). In case of TRF input parameters are always the same (weights for (*match*, *mismatch*, *indel*) = (2, 3, 5), *minimum alignment score* = 14, matching probability $P_m = 2$, indel probability $P_i = 10$), but the results are filtered to conform to *Our* program's parameters and to not contain indels. ATRs were looked for in Homo sapiens chromosome 8 genomic contig (sequence length: 1 291 612 bp, GenBank accession number: NT_023678) with different input parameters. Results are in Table III. We observed that generally *Our* method finds more ATRs than TRF and mreps. It was expected for TRF, as it is a heuristic method and it is assumed that it will only find some of the ATRs occurring in the input sequence, not all of them. As for mreps, *Our* method usually also finds more ATRs, mainly because mreps approach applies some methods to filter out found ATRs that are, according to the authors, not biologically significant. Because of this filtering, which cannot be disabled in the available tool, the output does not contain many ATRs that satisfy the assumptions. Additionally, mreps cannot adjust to the initial parameters (as *Our* algorithm does) and repeat once classified to has a *period* of a certain length is never treated

TABLE I
APPROXIMATE TANDEM REPEATS FOUND IN THE HOMO SAPIENS CLONE C05/443G COMPLEMENT RECEPTOR 1-LIKE (CR1L) GENE

| Start | End | Period | Exp. | Sequence |
|-------|-----|--------|------|---|
| 34 | 40 | 3 | 2.33 | acc acc a |
| 42 | 56 | 6 | 2.50 | ataatt ataatt tta |
| 49 | 61 | 5 | 2.60 | taatt ttatt taa |
| 69 | 85 | 5 | 3.40 | tette ttte ttcc tt |
| 78 | 89 | 5 | 2.40 | cttte ctcc ct |
| 81 | 100 | 8 | 2.50 | tectccc tectccc ttct |
| 99 | 116 | 8 | 2.25 | ctgectgc ctgectgc ct |
| 100 | 124 | 12 | 2.08 | tgectgectget tgecttctct t |
| 124 | 133 | 4 | 2.50 | ttgc ttgc tt |
| 127 | 144 | 8 | 2.25 | cttgettc ctctctct ct |
| 133 | 154 | 11 | 2.00 | tecttectcc tecttectcc |
| 138 | 184 | 12 | 3.92 | ctctctctctc ctctctctctc ctctctctctc ctctctctct |
| 142 | 192 | 16 | 3.19 | ctctctctctctctctc ctctctctctctctc ctctctctctctctct cct |
| 170 | 272 | 16 | 6.44 | ctctctctctctctct ctctctctctctctct ctctctctctctctct ctctctctctctctct ctctctctctctctct ctctctctctctctct |
| 264 | 278 | 6 | 2.50 | tttct tectct tat |
| 278 | 292 | 7 | 2.14 | tttctt tctctt t |
| 307 | 313 | 3 | 2.33 | acc acc a |
| 318 | 331 | 7 | 2.00 | aacattg aacattg |
| 350 | 359 | 5 | 2.00 | cttgt ctgt |

TABLE III
NUMBER OF ATRs FOUND IN NT_023678 SEQUENCE BY THREE DIFFERENT TOOLS, WITH VARIOUS PARAMETERS

| K^1 | $resolution^2$ | $Kprc^1$ | Parameters | | | | Results | | |
|-------|----------------|----------|------------|----------|----------|----------|---------|---------|---------|
| | | | $minPrd$ | $maxPrd$ | $minExp$ | $minLen$ | $Ours$ | $mreps$ | TRF^3 |
| 2 | 2 | 50 | 1 | 100 | 2 | 3 | 219 091 | 99 621 | 5 154 |
| 2 | 2 | 40 | 4 | 60 | 3 | 3 | 5 047 | 1 836 | 470 |
| 2 | 2 | 40 | 4 | 100 | 5 | 3 | 278 | 212 | 70 |
| 4 | 4 | 50 | 5 | 60 | 7 | 3 | 102 | 53 | 7 |
| 3 | 3 | 30 | 7 | 60 | 5 | 3 | 37 | 23 | 2 |

¹Parameter specific to *Our* tool. ²Parameter specific to *mreps* tool. ³Parameters for *TRF* tool are always the same, but the results are filtered according to $minPrd$, $maxPrd$ and $minExp$; indels are not allowed.

TABLE II
NUMBER OF ATRs FOUND IN FIVE DIFFERENT SEQUENCES
TOGETHER WITH PROGRAM RUNNING TIMES

| Accession number | Length [bp] | ATRs | Running time [ms] |
|------------------|-------------|-----------|-------------------|
| NT_023678 | 1 291 612 | 169 843 | 60 247 |
| NT_026446 | 5 594 590 | 712 107 | 275 742 |
| NT_011875 | 10 102 850 | 1 274 236 | 502 155 |
| NT_010859 | 15 400 898 | 1 967 766 | 770 234 |
| NT_009714 | 27 616 818 | 3 533 789 | 1 337 044 |

as one with a longer *period*.

Concluding, the paper presents a new approach for finding ATRs. The experiments performed demonstrate it is capable of finding, in a reasonable time, ATRs that comply with the requirements described by the input parameters. The results of quantitative comparison with different tools show that the algorithm is competitive with other available ATRs searchers and provides new possibilities for analysis of the occurrence of ATRs in the genomic sequences.

ACKNOWLEDGMENT

This work was supported by the European Union from the European Social Fund.

REFERENCES

- [1] R. Chakraborty, M. Kimmel, D. N. Stivers, L. J. Davison, and R. Deka, *Relative mutation rates at di-, tri-, and tetranucleotide microsatellite loci*, PNAS, Vol. 94, pp. 1041 A11046, 1997
- [2] S. Kruglyak, R. T. Durrett, M. D. Schug, and C. F. Aquadro, *Equilibrium distributions of microsatellite repeat length resulting from a balance between slippage events and point mutations*, PNAS, Vol. 95, pp. 1077410778, 1998
- [3] M. D. Vences, M. Legendre, M. Caldara, M. Hagihara, K. J. Verstrepen, *Unstable Tandem Repeats in Promoters Confer Transcriptional Evolvability*, Science 324, 1213 (2009)
- [4] C. T. McMurray, *Mechanisms of trinucleotide repeat instability during human development*, Nat Rev Genet. 2010 Nov; 11(11): 786-99.
- [5] A. J. Jeffreys, V. Wilson, S.L. Thein, *Individual-specific 'fingerprints' of human DNA*, Nature 316, 76-79, 1985
- [6] J. L. Weber and C. Wong, *Mutation of human short tandem repeats*, Hum. Mol. Genet. 2 (1993), pp. 1123-1128.
- [7] A. Merkel, N. Gemmell, *Detecting short tandem repeats from genome data: opening the software black box*, Brief. Bioinform. 9 (5) (2008) 355A1366.

- [8] R. Pokrzywa, *Application of the Burrows-Wheeler Transform for searching for tandem repeats in DNA sequences*, Int. J. Bioinf. Res. Appl. vol. 5, 432-446 (2009)
- [9] R. Pokrzywa, A. Polanski.: *BWtrs: A tool for searching for tandem repeats in DNA sequences based on the Burrows-Wheeler transform*, Genomics 96, 316-321 (2010)
- [10] M. Burrows, D.J. Wheeler, *A block-sorting lossless data compression algorithm*, SRC Research Report 124, Digital Equipment Corporation, California (1994)
- [11] P. Ferragina, G. Manzini, *Opportunistic data structures with applications*, In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, pp. 390-398, IEEE Computer Society Washington, DC, USA (2000)
- [12] S. Kurtz, J. V. Choudhuri, E. Ohlebusch, C. Schleiermacher, J. Stoye, R. Giegerich: *REPuter: The Manifold Applications of Repeat Analysis on a Genomic Scale*, Nucleic Acids Res., 29(22):4633-4642, 2001.
- [13] R. Kolpakov, G. Bana, G. Kucherov, *mreps: efficient and flexible detection of tandem repeats in DNA*, Nucleic Acids Research 31, 3672-3678 (2003)
- [14] G. Benson, *Tandem Repeats Finder: a program to analyze DNA sequences*, Nucleic Acids Research 27, 573-580 (1999)
- [15] Y. Wexler, Z. Yakhini, Y. Kashi, D. Geiger, *Finding Approximate Tandem Repeats in Genomic Sequences*, Journal of Computational Biology (2005) 928-942
- [16] D. Sokol, F. Atagun, *TRedD: A Database for Tandem Repeats over the Edit Distance*, Database (2010)
- [17] V. Boeva, M. Regnier, D. Papatsenko, V. Makeev, *Short fuzzy tandem repeats in genomic sequences, identification, and possible role in regulation of gene expression*, Bioinformatics (2006) 22 (6): 676-684
- [18] G. M. Landau, J. P. Schmidt, D. Sokol, *An Algorithm for Approximate Tandem Repeat*, Journal of Computational Biology, 8, 1-18, 2001