Journal of the ACS, Vol. 12, June.2021

A New Algorithm For Ssr Detection (Ssrs Finder)

Eng. Nada Hossam Sharkawy
Management Information System, Higher Institute of Computer and Information Technology, Cairo, Egypt

n.hossam@sha.edu.eg

Dr. Khaled El Menshawy
Management Information System, Higher Institute of Computer and Information Technology, Cairo, Egypt

dr.khalid.minshawi@sha.edu.eg

Abstract

One of the main challenges with bioinformatics software is that the size and complexity of datasets necessitate trading speed for accuracy, or completeness. To combat this problem of computational complexity, a plethora of heuristic algorithms have arisen that report a 'good enough' solution to biological questions. However, in instances such as Simple Sequence Repeats (SSRs), a 'good enough' solution may not accurately portray results in population genetics, phylogenetics and forensics, which require accurate SSRs to calculate intra- and inter-species interactions. And so, Biotechnologists took a lot of time to identify similar sequences on a DNA. In addition, the accuracy of the results was an important problem. In order to overcome this problem, biotechnologists need a modern hardware and equipment for the program to function efficiently. We proposed the SSRs finder application that detects the existing SSRs by determining the start and the stop, the min and the max and prepares a comparison between the old SSRs with the new SSRs then records the results and shows them in the quarry with high accuracy and speed.

***Key Words:* DNA, Microsatellites, NCBI, SSRs.**

1. INTRODUCTION

Since SSRs reveal characteristic functions of DNA replication, recombination and repair, they are important in studying biological systems interactions, as well as studying repeat expansion-based diseases with next-generation sequencing data. Many different approaches have been used to identify SSRs. They are based on k-mer decomposition, heuristic treatment, and phylogenetic trees.

k-mer decomposition technique focuses on decreasing memory usage and decreases execution time [1]. It is applied by storing only the newly found k-mer even if there is a minor difference between new k-mer and the preceding one. This leads to less accuracy compared to other methods and realistic phylogenetic relationship.

Although heuristic treatment implies more biologically relevant results, its execution time is high compared to other techniques [2]. The reason is that the heuristic techniques focus on detecting SSRs with respect to biological aspects i.e. single-nucleotide mutation.

Traditionally, the precise identification of SSRs from the DNA strand has been viewed as a substructure problem. There is a need to detect SSRs with less time, less memory, high accuracy, and biological relevant results. These features are real challenges for the researchers.

To overcome these problems, we introduce a new algorithm “SSRs Finder”. In this approach every M pattern in the S sequence is searched and the search is at a particular iteration and when the repeat is found again when looking for another repeat in the same process on the DNA strand it is compared in terms of number of iterations and takes the largest number of iterations and discards At least, all SSRs can be found in a single repeat across the DNA strand. Furthermore, there is no need for any post filtering because this method avoids over-selecting the same SSR that is commonly used in many methods, most notably with regular expression-based methods.

The results imply that “SSRs Finder” is efficient compared to other works. It detects SSRs with less time by 62s. The memory used is less than others by 600 MB. From the perspective of accuracy, “SSRs Finder” is more accurate compared to others.

The rest of the paper is structured as follows. Section 2 describes the related models in the literature and differences between them. Section 3 describes DNA sequences, how they work, and Simple Sequence Repeats (SSRs). Section 4 describes the proposed model. Afterwards, we review the performance measures, and discuss the experimental results of the proposed model in section 5. Finally, section 6 concludes the work and presents the future work.

2. RELATED WORK

In this section, some of other research works are discussed in detail, their algorithms, their features, and their weak points.

In 2001, Temnykh et al. introduced SSRIT system which detects SSRs [3]. The system applies detection with low accuracy which leads to imperfection in detection.

Another algorithm was introduced by Kolpakov et al. [2] for detection of SSRs (mreps) in 2003. It consists of two main parts: the first one (upper frame) collects certain repeated sequences through an efficient combinatorial algorithm. Those sequences serve as ‘raw material’ for the second part (lower frame), which applies to them an heuristic treatment in order to obtain biologically relevant repeats. The system implies good results however, the execution time and the memory usage are extremely high.

In 2011, Jighly et al. [4] introduced a new algorithm which is based on aligning sequences after determining the repeated units first, and the last SSR nucleotides positions.

This results in a shifting process according to the inserted repeated unit type. When studying the phylogenic relations before and after applying the new algorithm, many differences in the trees were obtained by increasing the SSR length and complexity. However, less distance between different linage had been observed after applying the new algorithm. The system performs results in a more realistic phylogenic relationship however, the computational complexity is very high.

Pickett et al. [1] introduced a new version of Kmer-SSR in 2017. It utilizes k-mer decomposition to provide an exhaustive or filtered approach to finding all SSRs in a given sequence. This version of k-mer decomposition works by identifying all subsequences of length ‘k’ while tracking the start position of each k-mer. K-mer lengths are defined by the user as the SSR period length. Kmer-SSR minimizes the usage of random access memory (RAM) by performing k-mer decomposition and only storing k-mers that are the same as the preceding k-mer (SSR period length). If a k-mer is not identical to a k-mer found k bases previously, the previously identified k-mers will be discarded and k-mer decomposition will occur for the rest of the sequence. This version implies less accuracy compared to previous research works even if the execution time is less than them.

Later, Avvaru et al. [5] introduced a new algorithm PERF. In the system, the user specified input file is read and an iterator is applied to access all the sequence records in the file serially. For each sequence, the identification of repeats is performed on a position specified by the user to detect all possible repeats of length specified by the user. Once the full repeat is identified, it is printed out along with the repeat class and other details such as the coordinates, motif length and total repeat length. The system implies high memory usage where the accuracy is comparable to others.

Some approaches, as discussed above, focus on the accuracy of the results, in contrast, the others’ main challenge is to perform less computation and smaller memory usage. For this point, applying a system that detects SSRs with high accuracy and less time and memory is a great challenge.

3. FUNDAMENTAL KNOWLEDGE

This section describes DNA and its structure in addition to Simple Sequence Repeats and their importance.

3.1 DNA Sequence

Deoxyribonucleic acid (DNA) is a molecule which carries genetic instructions for the development, functioning, growth and reproduction of all known organisms and many viruses [6]. It is used in many scientific fields which are genetic engineering, DNA profiling, DNA enzymes or catalytic DNA, bioinformatics, DNA nanotechnology, history and anthropology, and information storage.

DNA is composed of two polynucleotide chains that coil around each other to form a double helix. The two DNA strands are known as polynucleotides as they are composed of simpler monomeric units called nucleotides. Each nucleotide is composed of one of four nitrogen containing nucleobases (cytosine [C], guanine [G], adenine [A] or thymine [T]), a sugar called deoxyribose, and a phosphate group [7]. The nucleotides are joined to one another in a chain by covalent bonds (known as the phosphor-diester linkage) between the sugar of one nucleotide and the phosphate of the next, resulting in an alternating sugar-phosphate backbone [8].

The nitrogenous bases of the two separate polynucleotide strands are bound together, according to base pairing rules (A with T, C with G), with hydrogen bonds to make doublestranded DNA. The complementary nitrogenous bases are divided into two groups: pyrimidines and purines. In DNA, the pyrimidines are thymine and cytosine; the purines are adenine and guanine [9].

3.2 Simple Sequence Repeats (SSRs)

A Simple Sequence Repeat (microsatellite) is a tract of repetitive DNA in which certain DNA motifs (ranging in length from two to six or more base pairs) are repeated, typically 5–50 times [4][5]. Microsatellites occur at thousands of locations within an organism's genome. They have a higher mutation rate than other areas of DNA leading to high genetic diversity [1].

Microsatellites are often referred to as short tandem repeats (STRs) by forensic geneticists and in genetic genealogy, or as simple sequence repeats (SSRs) by plant geneticists. Microsatellites and their longer cousins, the minisatellites, together are classified as VNTR (variable number of tandem repeats) DNA. The name "satellite" DNA refers to the early observation that centrifugation of genomic DNA in a test tube separates a prominent layer of bulk DNA from accompanying "satellite" layers of repetitive DNA [10].

Fig.1 shows the occurrence of SSRs in DNA sequences. Simple Sequences Repeats are used extensively for DNA profiling in cancer diagnosis, in kinship analysis (especially paternity testing) and in forensic determination. It is also used in genetic association analysis to locate a gene or mutation responsible for a specific trait or disease and many viruses. Precision satellites are also used in population genetics to measure levels of association between subspecies, groups, and individuals [11].

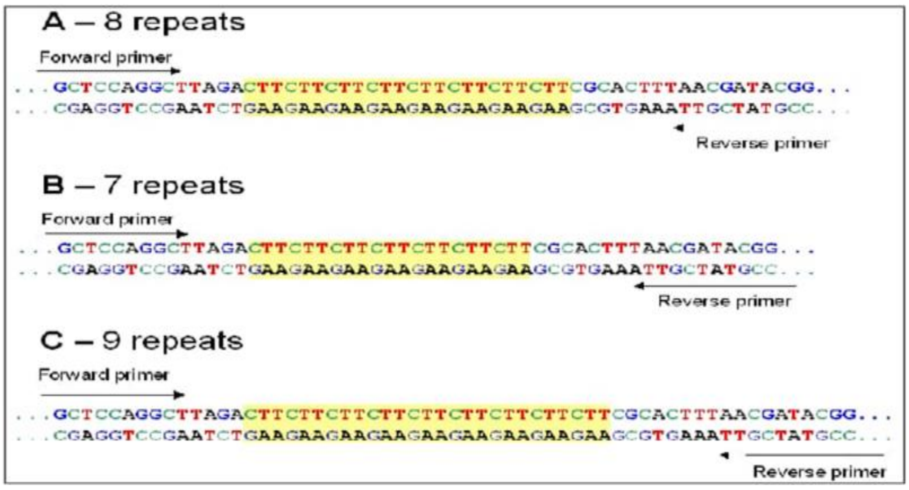
A – 8 repeats
Forward primer
...GCTCCAGGCTTAGACTTCTTCTTCTTCTTCTTCTTCTTCGCACTTTAACGATACGG...
...CGAGGTCCGAATCTGAAGAAGAAGAAGAAGAAGAAGAAGCGTGAAATTGCTATGCC...
Reverse primer
B – 7 repeats
Forward primer
...GCTCCAGGCTTAGACTTCTTCTTCTTCTTCTTCTTCGCACTTTAACGATACGG...
...CGAGGTCCGAATCTGAAGAAGAAGAAGAAGAAGAAGCGTGAAATTGCTATGCC...
Reverse primer
C – 9 repeats
Forward primer
...GCTCCAGGCTTAGACTTCTTCTTCTTCTTCTTCTTCTTCTTCGCACTTTAACGATACGG...
...CGAGGTCCGAATCTGAAGAAGAAGAAGAAGAAGAAGAAGAAGCGTGAAATTGCTATGCC...
Reverse primer

Figure 1. **Example of SSRs within a DNA sequence**

4. MATERIALS AND METHODS

In this paper, we compare our new algorithm for SSR alignment with the common alignment algorithms used in other programs. The new algorithm as shown in Fig.1 detects

SSRs by searching for them in the sequence with a specified repeat length in a given sequence and calculating the length of the SSRs. As a result, the SSR is saved in a list when the length exceeds 10 repeats.

The algorithm is applied according to the following major steps:

4.1 Sspecify Sequence and its Range. The user must specify:

4.1.1 The file which contains the sequence needed to detect its SSRs.

4.1.2 Search pane for SSRs whether on the whole gene or only part of it by specifying the start point and the stop point of searching nucleotides on the sequence (the start must be >= 1 and the stop must be <= length of the sequence).

4.2 Check Sequence Range. The system checks that the data file contains DNA sequence in addition to the start and stop points to ensure that start point doesn't lie before start of sequence and stop point doesn't exceed the end of the sequence. If it is found that the range contains a problem, the system informs the user.

4.3 Check for Presence of SSRs in Database. The system searches for the pre-entered gene with the same search pane and SSR length. If it is found in the database, the system ignores step 4 and 5 and performs step 6.

4.4 Determine the Simple Sequence Repeats. The system searches for the repeats in the specified range according to the following steps:

4.4.1 The system iterates on the sequence from start to stop – (min * 4) to detect SSRs to the last one can be found.

4.4.2 The system starts searching for SSRs at each nucleotide starting with min length of SSR till reaching max.

4.4.3 If the nucleotides is repeated, a counter starts counting the number of times the repeat is found till reaching the end of the repeat.

4.4.4 If the number of detecting of the repeat in follow is between 5 to 50 times, the repeat is recorded in a list with its position and length.

4.4.5 If more than one SSR is found in the same region, they are compared to each other and the largest SSR (in terms of number of repeats) is recorded.

4.5 Save the Determined SSRs. The system saves the list of SSRs with their gene ID, position on the gene, length of repeat, and the number of repetitions of the repeat in the database to be easily detected later.

4.6 Display the Results. The system displays SSRs to the user and their gene ID, position on the gene, length of repeat, and the number of repetitions of the repeat in a table.

5 EXPERIMENTS

We implemented the system by C# programming language on PC with Processor: Intel® Core™ i5-6200U CPU @ 2.30 GHz 2.40 GHz, RAM: 8.00 GB, Windows Version: Windows 10 Pro. The system is composed of desktop application and web application. The data is applied to the system for 400 times. The system is compared to other systems which are Kmer-SSR [1], MISA [4], MREPS [2], and SSRIT [10].

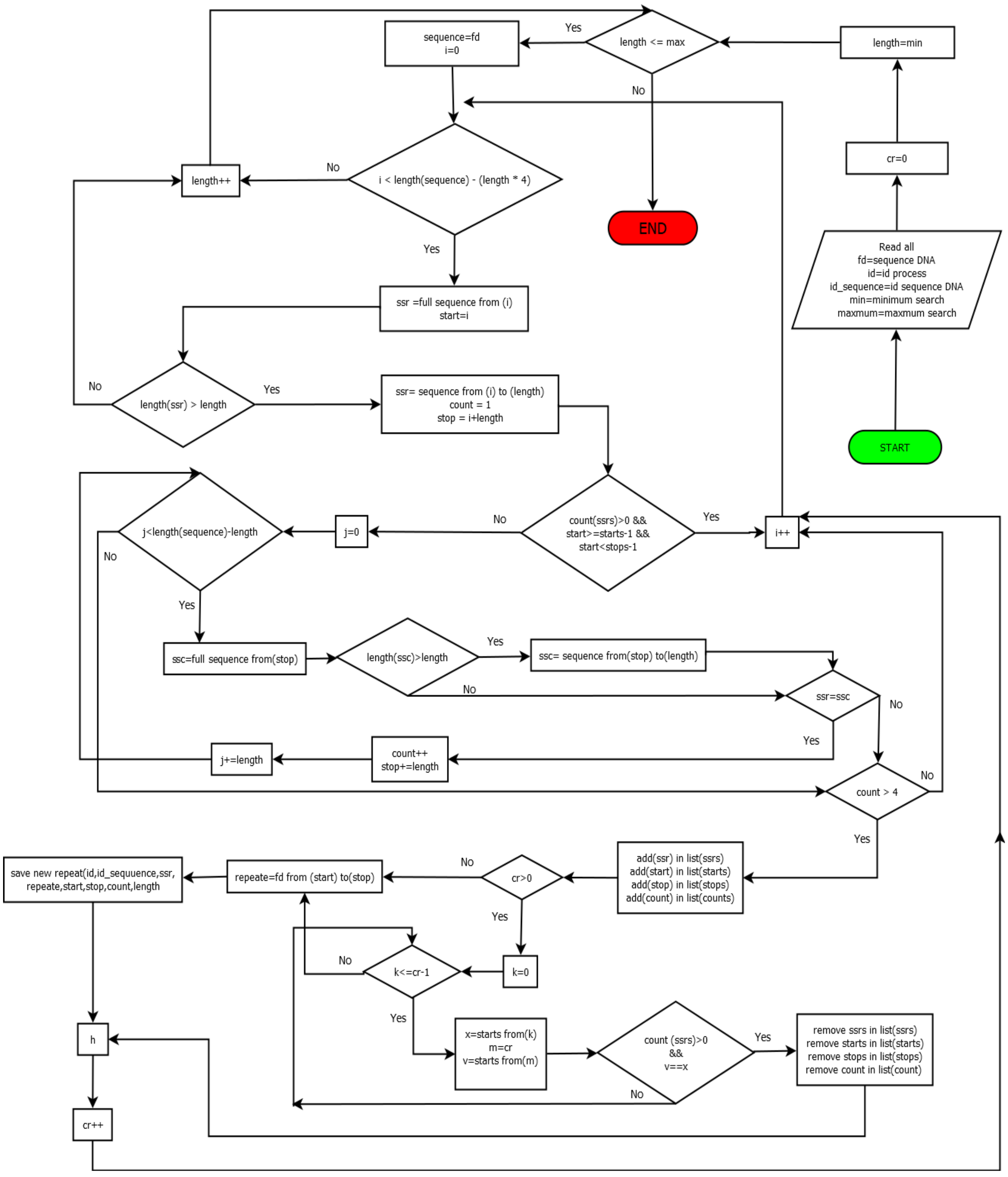

Figure 2. ***SSRs Finder Algorithm***

5.1 Data Gathering

The data needed for testing the system can be obtained from various websites, such as: evoltree [12], SSRome [13], Sequence Read Archive [14], and NCBI [15]. We obtained the data from NCBI website. The file format is FASTA files (.FASTA). the file begins with a single-line description, followed by lines of sequence data. The description line begins with a greater-than (“>”). Sequence file in FASTA format can contain several sequences. This is an example of ..FASTA file with single sequence:

```
>NG_016623.1 Homo sapiens syntax in binding protein 1 (STXBP1), RefSeqGene on chromosome 9
AAAAAAAAAAAATCCATTGCTGGCCTGGCGCAGTGGCTCATGCCTGAAATCCC
AGCACTTTGGGAAGCCGAGGCGGGAGGATCAGTTGAGGTCAGGAGTTTGAGAC
CAGCCTAGCCAACCACATGGTGAAACCCTGACTCTACTGAAAATACAAAAATT
AGCCAGGTGGCACACACCTGTAATCCCAGCTACTCGGGAGGCTGAGGCAGGAG
AATCGGTTGAACCCGGGAGGCAGAGCTTGCAGTGAGTAGAGATCATGCTTCTG
CACTCCAGCCTGGGCGACAGAGTGAGACTCTGACTCAAAAAAAAAAAAAAA
AAATCCATTGCAACACCTAAACATTGGGAGAT
```

We downloaded files with single sequence. We selected Oryza sativa genes and the number of sequences we collected for this gene was 400 and extracted SSRs for each of its sequences.

5.2 Evaluation Criteria

In this paper, the system is evaluated using system features analysis, accuracy, and runtime which will be described in this section.

5.2.1 System Features Analysis

System features differs from each system. To ensure the evaluation, we compared the features in the systems in the comparison. We check:

5.2.1.1 Operating Features including the operating system the system runs on, the programming language used to implement the system, and applying multi-threading.

5.2.1.2 Input-Output Features including the input format of the data files, the output format of the result, and if the system prepares analysis report on the usage of the algorithm monthly.

5.2.1.3 Functionality Features including detecting perfect repeats (which is repeated without any single mutation), detecting partial repeats (which the repeat is not completed in the last one), detecting imperfect repeats (which is repeated with single mutation), detecting length cut-off (the length of the SSR), detecting unit cut-off (the length of one repeat in the SSR), and search specific SSRs.

5.2.2 Accuracy

We check the accuracy of detection in two parts: The accuracy of the SSR detected, and the accuracy of the number of SSRs detected.

5.2.3 Runtime and Memory Usage

One of the main sections to check efficiency is Runtime and Memory Usage. The reason is that we need the results in less time and memory due to the huge amount of data. We check the time needed to detect the SSRs and the memory needed to save the results using custom time measurement and memory measurement codes.

5.3 Results

The results of the evaluation of the system will be described in this section.

5.3.1 System Features Analysis

Tables 1-3 show the results of the comparison between our proposed algorithm and previous studies with respect to system features. According to the tables, more than half of the features were applied in our algorithm which exceeds number of features of previous algorithms. On the other hand, MREPS was the only application which applied partial repeats and imperfect repeats. All the algorithms applied single-threading while Kmer-SSR.

TABLE 1. SYSTEM FEATURES ANALYSIS (OPERATING FEATURES)

Program	Comparisons		
	Operating System	Language	Multi- threaded
MREPS	Linux	C	○
SSRIT	Linux	Perl	○
SSRs Finder	windows	C#	○

is applied ○ is not applied •

TABLE 2. SYSTEM FEATURES ANALYSIS (INPUT-OUTPUT FEATURES)

Program	Comparisons		
	Input Format	Output Format	Analysis Report
MREPS	FASTA	Text	○
SSRIT	FASTA	TSV	○
SSRs Finder	FASTA	Text	•

is applied ○ is not applied •

TABLE 3. SYSTEM FEATURES ANALYSIS (FUNCTIONALITY FEATURES)

Program	Comparisons						
	Exhaustive	Perfect Repeats	Partial Repeats	Imperfect Repeats	Length cut-off	Unit cut-off	Search Specific SSRs
MREPS	○	•	•	•	•	•	○
SSRIT	○	•	○	○	○	•	○
SSRs Finder	•	•	○	○	•	•	•

is applied ○ is not applied •

5.3.2 Accuracy

The accuracy of various tools was assayed using two simulated datasets where the results are in Table 4. On both the simulated datasets, SSRs Finder was the only tool which reported 100% of the expected SSRs. By applying the same sequence to the SSRIT program, we found that it explored SSRs with an approximate rate of 85%, and by comparing the results after executing the sequence on the MREPS program, we noticed that it explored SSRs by 75%, with an error rate.

TABLE 4. EVALUATION OF THE ACCURACY AND THE NUMBER OF SSRS REPORTED BY OF VARIOUS TOOLS ON RICE CHROMOSOME 1

Program	Comparisons		
	Total SSRs reported	SSRs in range (a)	Detected SSRs in range % (b)
MREPS	266	545	48.8
SSRIT	515	545	94.49
SSRs Finder	545	545	100

a. SSRs in range indicate the number of identified repeats that match the evaluation criteria
b. The percentage calculated in relation to the number of repetitions specified by the programs used.

5.3.3 Runtime and Memory Usage

We measured time and memory needed by adding a custom time measurement and memory measurement codes in Table 5. The time indicates average of 5 independent runs in seconds. Memory usage is an average of 3 independent runs in megabytes. Despite being 100% comprehensive algorithm, SSRs Finder was faster than all the tested tools.

TABLE 5. EVALUATION OF THE RUNTIME AND MEMORY USAGE OF VARIOUS TOOLS ON RICE CHROMOSOME 1

Program	Comparisons	
	Time(s)	Memory usage (MB)
MREPS	84.33	2411.20
SSRIT	120.67	778.74
SSRs Finder	22.156	164

6 CONCLUSION

With the addition of new genomes every day, efficient and comprehensive identification of microsatellites from DNA sequences is crucial for researchers studying these important elements. In spite of a wide gamut of existing tools, users currently have to make a trade-off between speed and exhaustiveness while choosing a tool that suits their requirement. We developed an algorithm that addresses both these challenges. Even when run on a computer with modest specifications, SSRs Finder takes less than 2 min to accurately identify all microsatellites from one of the largest input files—the human genome. SSRs Finder can be installed effortlessly and is easy to use; the only required parameter is the input file in FASTA format. At the same time, all search parameters can be configured by the user at runtime via a clean command-line interface, thereby offering full flexibility and control on the output. Though the evaluation reported here is performed using typical definition of SSRs (2–6 nt long motifs), the algorithm does not discriminate micro- and minisatellites, and can be used to identify repeating motifs of any length. In addition, SSRs Finder is OS-agnostic, and works the same on any system.

REFERENCES

[1] B. D. Pickett, J. B. Miller and P. G. Ridge, "Kmer-SSR: a fast and exhaustive SSR search algorithm," 2017.

[2] R. Kolpakov, G. Bana and G. Kucherov, "mreps: efficient and flexible detection of tandem repeats in DNA," 2003.

[3] A. Jighly, A. Hamwieh and F. C. Ogbonnaya, "Optimization of sequence alignment for simple sequence repeat regions," 2011.

[4] A. K. Avvaru, D. T. Sowpati and R. K. Mishra, "PERF: an exhaustive algorithm for ultra-fast and efficient identification of microsatellites from large DNA sequences," 2017.

[5] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts and P. Walter, Molecular Biology of the Cell.

[6] H. Lodish, A. Berk, S. L. Zipursky, P. Matsudaira, D. Baltimore and J. Darnell, Molecular Cell Biology.

[7] C. Molnar and J. Gair, Concepts of Biology.

[8] G.-F. Richard, A. Kerrest and B. Dujon, "Comparative Genomics and Molecular Dynamics of DNA Repeats in Eukaryotes," 2020.

[9] X. Gou, H. Shi, S. Yu, Z. Wang, C. Li, S. Liu, J. Ma, G. Chen, T. Liu and Y. Liu, "SSRMMD: A Rapid and Accurate Algorithm for Mining SSR Feature Loci and Candidate Polymorphic SSRs Based on Assembled Sequences," 2020.

[10] B. Budowle, S. Schutzer and S. Morse, "Ricin forensics," in Microbial Forensics, 2020.

[11] Evoltree. https://www.evoltree.eu/index.php/e-resources/databases/ssr

[12] SSRome. http://mggm-lab.easyomics.org/

[13] Sequence Read Archive. http://www.ncbi.nlm.nih.gov/Traces/sra/

[14] National Center for Biotechnology Information (NCBI). http://www.ncbi.nlm.nih.gov/Traces/sra/

[15] https://archive.gramene.org/db/markers/ssrtool