Journal of Theoretical and Applied Information Technology
30th September 2016. Vol.91. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EXTENDED PERFECT AND IMPERFECT REPEATS FINDER USING MOLECULAR DNA SEQUENCES

[1] DR. RAJU BHUKYA,
[1] Department of Computer Science and Engineering, National Institute of Technology, Warangal, INDIA
E-mail: [1]drrajunitw@gmail.com

ABSTRACT

Microsatellites are ubiquitous short tandem repeats found in all known genomes and are known to play a very important role in various studies and fields including DNA fingerprinting, paternity studies, evolutionary studies, virulence and adaptation of certain bacteria and viruses etc. Therefore, it is of importance to study distribution, enrichment and polymorphism of microsatellites in the genomes of interest. For this, the prerequisite is the availability of a computational tool for extraction of microsatellites (perfect as well as imperfect) and their related information from whole genome sequences. Examination of available tools revealed certain lacunae in them and prompted us to develop a new tool.

Keywords: *DNA Sequence, Microsatellites, Perfect, Imperfect, Extension of repeats.*

1. INTRODUCTION

Bioinformatics is a multi-disciplinary science that uses methods and principle from mathematics, computer science and statistics for analyzing biological data. DNA sequencing plays a key role in various applications in computational biology for data analysis like feature extraction, searching, disease and structural analysis. Microsatellites or simple sequence repeats (SSRs) are the nucleotide sequences arising out of tandem repeating of short sequence motifs of the size 1–6bp [18]. Microsatellites have been found in all the known genomes so far and are widely distributed both in coding and non-coding regions [1], [2]. This paper deals with repeated perfect and imperfect repeats repeating more than 10bp in number. Mutations occurring at microsatellite loci within or near certain genes have been implicated to be responsible for some human neurodegenerative diseases [18]. Furthermore, microsatellite instability has also been implicated in the induction of cancer [19]. Owing to their high mutability, it is thought that the microsatellites are one of the sources of genetic diversity [10]. Imperfect microsatellites are more stable than perfect microsatellites as they are less prone to slippage mutations [17] and are known to play a role in gene regulation [14]. A large body of microsatellite data from several genome sequences still remains unexplored. Studies pertaining to distribution, enrichment, mutational dynamics of microsatellites along with their role in gene function and expression are very essential to understand the processes that underpin the evolution and diversity of genomes. However, a large body of microsatellite data from several genome sequences still remains unexplored. Studies pertaining to distribution, enrichment, mutational dynamics of microsatellites along with their role in gene function and expression are very essential to understand the processes that underpin the evolution and diversity of genomes.

We made a survey of existing software tools for identification and extraction of microsatellites from nucleotide sequences. these tools can be divided into the two groups: those which can identify only perfect microsatellites (e.g. SSRF [20], Poly [5], SSRIT [21]) and the others which can identify perfect as well as imperfect microsatellite (which can identify perfect as well as imperfect microsatellites (e.g. TRF [4], ATR Hunter [22], 2004) , Sputnik [23], and IMEx [16]. Our survey also revealed certain 'lacunae' in the tools. Programs such as 'mreps' [11] and TandemSWAN [7] consider only substitutions but not indels. The algorithms of TRF [4], ATR Hunter [22] and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

STRING [15] have been designed to find tandem repeats of large-size motifs as large as 2000 bases and hence large numbers of microsatellites go unidentified by these methods. Programs like TRbase [6] a database for tandem repeats in disease genes. Tools like TROLL [8] generates perfect and imperfect repeats, special programs like [9] for simple sequences with complex evolution, to know the differential distribution of repeats [12], microsatellite with in gene structure [13] for the analysis of gene structure. Many of these programs, [9] do not generate alignments between perfect and imperfect microsatellites. We develop new algorithm, which is fast, highly sensitive and also flexible where user can set the limit of imperfection (for perfect microsatellite and imperfect microsatellite both). The output comprises of a list of microsatellite, each of which with information content, sequences alignments (starting and ending point).

Perfect Repeat:

In a given sequence, a tandem repeat of a size *n* a given sequence, a perfect repeat of a size *n* is a subsequence which repeats continuously twice or more in the sequence (specified by the user). DNA molecules are subject to a variety of mutational events. One of the less well understood is perfect repeat duplication in which a stretch of DNA, which we call the pattern, is converted into two or more copies, each following the preceding one in a contiguous fashion. For example we could have … TCGGA … → … TCGGCGGCGGA … in which the single occurrence of triplet CGG has been transformed into three identical, adjacent copies. So here, CGG is the perfect.

TACGAGT**ACGGCGGCGG**ATGCCGTAT

Figure.1. This is three consecutive occurrence of the pattern 'CGG'.

In a given sequence, after certain intervening nucleotides, the repeat motif does not contain any imperfection (i.e. *k*=0).

TACGAGTA**CGG**CAG**CGG**ATGCCGT

Figure 2. Here, nucleation sites characterized by two identical motifs inverted by 3 nucleotides. The intervening CAG is an iteration of CGG with c→ G operation (k=1).

Imperfect nucleotides:

In a given sequence, imperfect repeat is the extension of the nucleation sites of the motif (with imperfection less then '*k*' value) as long as some termination criteria is satisfied. The number of imperfections between the individual repeat copy and the perfect repeat motif is more than the limit (denoted by '*k'* parameter set by the user) and (ii) the percentage of imperfection is more than the limit set by the user (denoted by '*p'* parameter). The percentage imperfection is calculated as follows:

$$p = \frac{\text{number of point mutations in the observed tract}}{\text{total number of bases in the equivalent perfect tract}} X100$$

The user can set the value of '*k'* between 0 to '*m*' where *m* is the repeat motif size. Once the termination criteria is satisfied, only those candidate microsatellites that are more than the minimum repeat number of that repeat size set by the user (denoted by '*n*' parameter) are reported.

2. METHODS

2.1 Discovering perfect repeats (Exact Repeats):

The process to discover perfect Repeats in the given sequence file consists of two phases:

Initially start by looking for a subsequence of length one (mono nucleotide) and check for its continuous repetition. If there is a repetition of repeats, then increase the value of count for every repeats till the same repeat is found and if the count value is equal or greater than the user value (denoted by the '*n*'), then write the sequences into the file for every repeat. Follow the procedure for the next subsequence from the character just after the ending index. If a subsequence of length one (mono nucleotides) does not repeat continuously (given value by the user for mono nucleotides) up to end of the file, then increase the length of the subsequence (means Di nucleotides) and search for the repetition from starting point of the file to end of the file. Repeat the same procedure up to deca nucleotide. Always compare the count value to the user specified value if the count value is greater than or equal to then write the sequence into the file otherwise the algorithm goes for the next sequence and so on.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2.2 Discovering imperfect Repeats (Approximate perfect repeats):

The process to discover approximate perfect repeats (imperfect repeats) in the given sequence file.

(i) The no of imperfections between the individual repeat copy and the perfect repeat motif is more then, the limit (denoted by the '*k*' parameter) then we can write into the file for that repeat.

(ii) The percentage of imperfection is calculated for every repeats. If the percentage of imperfection is less than the limit set by the user (denoted by '*p*' parameter) then we can write into the file for that repeat.

Initially start by looking for a subsequences of length one (mono nucleotide) and check for its continuous repetition with *k*-mismatched ('*k*' value set by the user 1,2,3....). If there is a repetition of repeats, then look for the index till where the subsequences repeats itself continuously with *k*-mismatched. Here we compare the count value with user value (specified by the user '*n*'). For finding imperfect repeats, use normal string matching algorithm. The algorithm stores starting & ending indices for mono-imperfect repeats in a file and follow the procedure for the next subsequences from the character just after the ending index. If a subsequence of length mono does not repeat continuously, then increase the length of the subsequence to di-nucleotides and search for the repetition of all subsequences of DNA file starting point to end point of the DNA file and so on.

The process of finding exact compound tandem repeats is depicted in the Figure 3.

3. RESULTS:

Discovering perfect and imperfect repeats of proposed algorithm technique are implemented in Python programming language. For experiment, we used genome sequences for discovering perfect as well as imperfect repeats. The proposed algorithm finds the perfect repeat which is able to discover up to 20 or more in size. Here the technique which is used is simple string matching algorithm for finding perfect and imperfect repeats. To discuss the capabilities of our code, we analyzed the human atrophin1 gene (BC051795) and compared the result obtained with those obtained using Tandem Repeat Finder (TRF) [4] and Sputnik [23] and IMEx [16]. TRF was initially tested with the parameters used in the earlier studies [2], [6], [24] which yielded very few microsatellites. Hence, we used the most relaxed set of parameters (Match: +2, Substitution:-7, Indel: -7, min Score: 2) which yielded substantial number of microsatellites. This is because using TRF, the length of the microsatellite

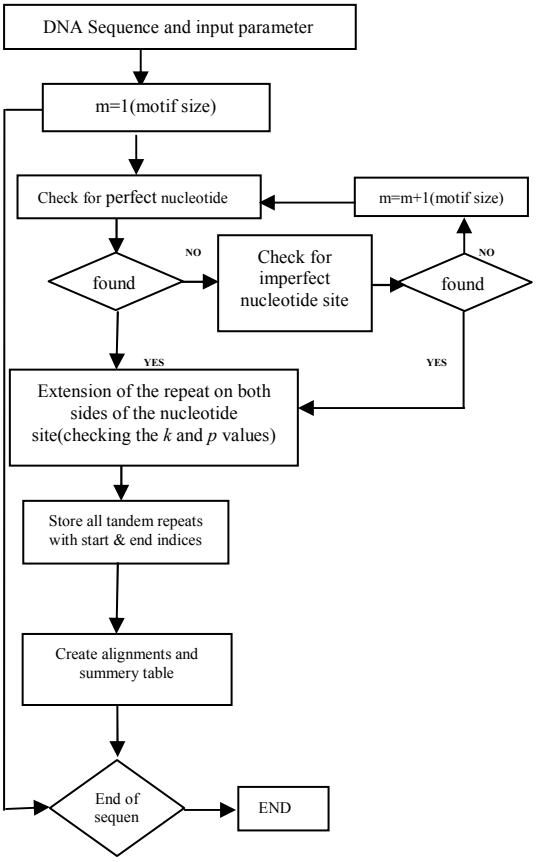

Figure 3: Process for Exact Compound Tandem Repeats

we used the least stringent parameters (Match: +1, Mismatch: -3, Min Score: -5), and for IMEx [16], we set the '*p*' value of all tracts to 10%; '*k*' value for each pattern size: Mono: 1, Di: 1, Tri: 1, Tetra: 2, Penta: 2, Hexa: 3 and further restricted to report only those microsatellites with minimum repeat copy number (Mono:5, Di: 3, Tri: 2, Tetra: 2, Penta: 2, Hexa: 2) to match those reported by TRF [4] and Sputnik [23]. TRF [4] and Sputnik [23] identified 50 and 19 repeats respectively, whereas IMEx [16] identified 146 microsatellite tracts. In our program we take the input as same as IMEx [16] but the program restricted to report only those microsatellite with minimum repeat copy cumber (Mono:5, Di: 3, Tri: 2, Tetra: 2, Penta: 2, Hexa: 2,Octa: 2, Enea: 2, Deca:2) to match those reported by TRF [4], Sputnik [23] and IMEx [16]. IMEx

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

gives the result up to size hexa but our program gives the up to deca nucleotide and more. The result is more than the IMEx in number of patterns which is shown in below Table.1.

Table.1:Microsatellite identified by EPI repeats in the human atrophin 1 gene(4382 bp). Newly found tracts identified as bold.

S.No	String	Sindex	Eindex	Repeats	Strings
1	A	509	514	6	AAAAAA
2	C	643	647	5	CCCCC
3	C	846	851	6	CCCCCC
4	**C**	**#888**	**892**	**5**	**CCCCC(new found)**
5	G	912	916	5	GGGGG
6	C	939	943	5	CCCCC
7	G	956	961	6	GGGGGG
8	G	980	985	6	GGGGGG
9	G	993	997	5	GGGGG
10	C	1007	1012	6	CCCCCC
11	C	1059	1063	5	CCCCC
12	C	1130	1134	5	CCCCC
13	C	1155	1159	5	CCCCC
14	C	1197	1201	5	CCCCC
15	G	1208	1212	5	GGGGG
16	C	1353	1358	6	CCCCCC
17	C	1438	1442	5	CCCCC
18	C	1472	1476	5	CCCCC
19	C	1488	1492	5	CCCCC
20	C	1518	1523	6	CCCCCC
21	C	1572	1576	5	CCCCC
22	C	1776	1780	5	CCCCC
23	C	1899	1903	5	CCCCC
24	C	2157	2161	5	CCCCC
25	G	2189	2193	5	GGGGG
26	C	2458	2462	5	CCCCC
27	C	2475	2479	5	CCCCC
28	C	2496	2500	5	CCCCC
29	C	2868	2873	6	CCCCCC
30	G	2975	2979	5	GGGGG
31	G	3174	3178	5	GGGGG
32	C	3494	3498	5	CCCCC
33	C	3554	3558	5	CCCCC
34	C	3874	3878	5	CCCCC
35	C	3884	3889	6	CCCCCC
36	G	3966	3970	5	GGGGG
37	C	4082	4087	6	CCCCCC
38	C	4100	4104	5	CCCCC
39	C	4231	4235	5	CCCCC
40	A	4308	4314	7	AAAAAAA
41	A	4339	4343	5	AAAAA
42	A	4361	4382	22	AAAAAAAAAAAA
43	AG	479	484	3	AGAGAG
44	AA	509	514	3	AAAAAA
45	CC	846	851	3	CCCCCC
46	GG	956	961	3	GGGGGG
47	GG	980	985	3	GGGGGG
48	CC	1007	1012	3	CCCCCC
49	**CT**	**#1319**	**1324**	**3**	**CTCTCT(new found)**
50	CC	1353	1358	3	CCCCCC
51	CT	1500	1505	3	CTCTCT
52	CC	1518	1523	3	CCCCCC
53	TC	1539	1544	3	TCTCTC
54	TG	1556	1561	3	TGTGTG
55	AG	2173	2178	3	AGAGAG
56	GC	2581	2588	4	GCGCGCGC
57	**GC**	**#2682**	**2687**	**3**	**GCGCGC(new found)**
58	CC	2868	2873	3	CCCCCC
59	AG	3837	3842	3	AGAGAG
60	CC	3884	3889	3	CCCCCC
61	AG	3959	3964	3	AGAGAG
62	**TC**	**#4068**	**4073**	**3**	**TCTCTC(new found)**
63	CC	4082	4087	3	CCCCCC
64	TG	4221	4226	3	TGTGTG
65	**GC**	**#4226**	**4231**	**3**	**GCGCGC(new found)**
66	TA	4278	4287	5	TATATATATA
67	AA	4308	4313	3	AAAAAA
68	AA	4361	4382	11	AA by 11 times
69	GGA	35	40	2	GGAGGA
70	**GAG**	**#219**	**224**	**2**	**GAGGAG(new found)**
71	**GAA**	**#244**	**249**	**2**	**GAAGAA(new found)**
72	GAG	284	289	2	GAGGAG
73	GAA	296	301	2	GAAGAA
74	GAA	321	326	2	GAAGAA
75	CAG	371	376	2	CAGCAG
76	AAG	411	416	2	AAGAAG
77	CAA	449	454	2	CAACAA
78	GAG	468	473	2	GAGGAG
79	GAG	492	497	2	GAGGAG
80	AAA	509	514	2	AAAAAA
81	ATG	589	597	3	ATGATGATG
82	GCA	598	603	2	GCAGCA
83	TCT	690	695	2	TCTTCT
84	ACC	724	729	2	ACCACC
85	TCC	743	748	2	TCCTCC
86	CCC	846	851	2	CCCCCC
87	CTC	871	876	2	CTCCTC
88	TGG	920	925	2	TGGTGG
89	GGG	956	961	2	GGGGGG
90	TCA	972	977	2	TCATCA
91	GGG	980	985	2	GGGGGG
92	AGC	1000	1005	2	AGCAGC
93	CCC	1007	1012	2	CCCCCC
94	ACT	1017	1022	2	ACTACT
95	GTG	1051	1056	2	GTGGTG
96	CAA	1063	1068	2	CAACAA
97	GCC	1070	1075	2	GCCGCC
98	GGT	1089	1094	2	GGTGGT
99	CCA	1113	1121	3	CCACCACCA
100	CAA	1178	1183	2	CAACAA
101	TCC	1274	1279	2	TCCTCC
102	CCT	1326	1331	2	CCTCCT
103	**TCT**	**#1338**	**1343**	**2**	**TCTTCT(new found)**
104	CCC	1353	1358	2	CCCCCC
105	TAG	1382	1390	3	TAGTAGTAG
106	GCA	1395	1400	2	GCAGCA
107	CCT	1402	1407	2	CCTCCT
108	TTC	1415	1420	2	TTCTTC
109	**TCC**	**#1419**	**1424**	**2**	**TCCTCC(new found)**
110	**CAA**	**#1492**	**1497**	**2**	**CAACAA(new found)**
111	CCC	1518	1523	2	CCCCCC
112	**CCA**	**#1575**	**1580**	**2**	**CCACCA(new found)**
113	**CCT**	**#1581**	**1586**	**2**	**CCTCCT(new found)**
114	GCC	1594	1599	2	GCCGCC
115	CCA	1668	1673	2	CCACCA
116	CAC	1692	1697	2	CACCAC
117	CAG	1710	1757	16	CAG 16 times
118	**CCT**	**#1779**	**1787**	**3**	**CCTCCTCCT(new found)**
119	CCA	1823	1828	2	CCACCA
120	**CCA**	**#1878**	**1883**	**2**	**CCACCA(new found)**
121	CAC	2072	2077	2	CACCAC
122	CAG	2125	2130	2	CAGCAG
123	CCT	2239	2244	2	CCTCCT
124	CCA	2376	2384	3	CCACCACCA
125	GCC	2405	2410	2	GCCGCC
126	GAG	2445	2450	2	GAGGAG
127	GGT	2507	2512	2	GGTGGT
128	CAA	2549	2554	2	CAACAA
129	AAG	2631	2636	2	AAGAAG
130	TGG	2647	2652	2	TGGTGG
131	GCG	2660	2665	2	GCGGCG
132	**AGC**	**#2674**	**2679**	**2**	**AGCAGC(new found)**
133	**GAA**	**#2688**	**2693**	**2**	**GAAGAA(new found)**
134	AGG	2779	2784	2	AGGAGG
135	CCC	2868	2873	2	CCCCCC
136	CAG	3023	3028	2	CAGCAG
137	GCA	3326	3331	2	GCAGCA
138	GGC	3344	3349	2	GGCGGC
139	CAC	3402	3407	2	CACCAC
140	TTC	3589	3594	2	TTCTTC
141	TGC	3608	3613	2	TGCTGC
142	CAG	3655	3660	2	CAGCAG
143	TCA	3662	3667	2	TCATCA
144	CAG	3720	3728	3	CAGCAGCAG
145	AGG	3775	3780	2	AGGAGG
146	ACT	3781	3786	2	ACTACT
147	GAA	3797	3802	2	GAAGAA
148	CCC	3884	3889	2	CCCCCC
149	TGC	3926	3931	2	TGCTGC
150	**GTG**	**#4003**	**4008**	**2**	**GTGGTG(new found)**
151	CCC	4082	4087	2	CCCCCC
152	**CTC**	**#4095**	**4100**	**2**	**CTCCTC(new found)**
153	GCC	4121	4129	3	GCCGCCGCC
154	ATT	4152	4157	2	ATTATT
155	TAA	4270	4275	2	TAATAA
156	AAA	4308	4313	2	AAAAAA
157	AAA	4361	4381	7	AAA 7 times
158	**AAGA**	**#297**	**304**	**2**	**AAGAAAGA(new found)**
159	**CTCC**	**#550**	**557**	**2**	**CTCCTCC(new found)**
160	**GCCC**	**#714**	**721**	**2**	**GCCCGCCC(new found)**
161	**CCAC**	**#725**	**732**	**2**	**CCACCCAC(new found)**
162	**CCCA**	**#1010**	**1017**	**2**	**CCCACCCA(new found)**
163	GTGG	1086	1093	2	GTGGGTGG
164	ACCT	1099	1106	2	ACCTACCT
165	TCCC	1483	1490	2	TCCCTCCC
166	TCCC	1633	1640	2	TCCCTCCC
167	CCCA	1663	1670	2	CCCACCCA
168	CCCA	1798	1805	2	CCCACCCA
169	**CTCC**	**#1849**	**1856**	**2**	**CTCCCTCC(new found)**
170	CCTC	2235	2242	2	CCTCCCTC
171	CAGT	2531	2538	2	CAGTCAGT
172	**AACA**	**2550**	**2557**	**2**	**AACAAACA(new found)**
173	GCGC	2581	2588	2	GCGCGCGC
174	AGCG	2677	2684	2	AGCGAGCG
175	CCAT	2954	2961	2	CCATCCAT
176	GCAC	3680	3687	2	GCACGCAC
177	**CTGC**	**3765**	**3772**	**2**	**CTGCCTGC(new found)**
178	**GGGA**	**#3968**	**3979**	**3**	**GGGA 3 times (new found)**
179	**GACA**	**#3978**	**3985**	**2**	**GACAGACA(new found)**
180	TATA	4278	4285	2	TATATATA
181	**CCAA**	**#4319**	**4326**	**2**	**CCAACCAA(new found)**
182	AACC	4327	4334	2	AACCAACC
183	CAAA	4334	4341	2	CAAACAAA
184	AAAA	4361	4380	5	AAAA 5 times
185	GGGCC	338	347	2	GGGCCGGGCC
186	**CCTGG**	**#1200**	**1209**	**2**	**CCTGGCCTGG(new)**
87	**TCCCA**	**#1543**	**1552**	**2**	**TCCCATCCCA(new)**
188	CCCCT	2012	2021	2	CCCCTCCCCT
189	CCACC	2206	2215	2	CCACCCCACC
190	TCATG	2927	2936	2	TCATGTCATG
191	AGCTG	3692	3701	2	AGCTGAGCTG
192	**CCCCT**	**#4128**	**4137**	**2**	**CCCCTCCCCT(new)**
193	AAAAA	4361	4380	4	AAAAA 4 times
194	TGACTC	677	688	2	TGACTCTGACTC
195	CCCCTC	883	894	2	CCCCTCCCCCTC
196	**TCCTCT**	**#1422**	**1433**	**2**	**TCCTCT(new found)**
197	CATCAC	1683	1694	2	CATCACCATCAC
198	CAGCAA	1698	1709	2	CAGCAACAGCAA
199	CAGCAG	1710	1757	8	CAGCAG 8 times
200	CTCTTC	1955	1966	2	CTCTTCCTCTTC
201	CTCTTC	1970	1981	2	CTCTTCCTCTTC
202	GAGCGC	2697	2708	2	GAGCGCGAGCGC

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

203	GAGCGC	2727	2738	2 GAGCGCGAGCGC
204	GCACCT	3428	3439	2 GCACCTGCACCT
205	AAGGCC	3987	3998	2 AAGGCCAAGGCC
206	AACCAA	4321	4332	2 AACCAAAACCAA
207	AAAAAA	4361	4378	3 AAAAA 3 times
208	CCCTTTC	3212	3225	2 CCCTTTCCCCTTTC
209	TGCCCCT	4247	4260	2 TGCCCCTTGCCCCT
210	AAAAAAA	4361	4381	3 AAAAAA 3 times
211	CTTCCCAG	1442	1457	2 CTTCCCAGCTTCCCAG
212	AAAAAAAA	4361	4376	2 AAAAAAAAAAAAAAAA
213	CAGCAGCAG	1710	1754	5 CAGCAGCAG 5 times
214	AAAAAAAAA	4361	4378	2 9 As 2 times
215	AAAAAAAAAA	4361	4380	2 10 As 2 times

As we can see from the result, our program reports many more tracts which are missed by the other three program (TFR [4], Sputnik [23], IMEx [16]). It is important to mention that Sputnik does not report the mononucleotides and after that the penta nucleotides(i.e. hexa nucleotide and more) and IMEx also not report the some of the repeats size mononucleotide to hexa nucleotides and it does not report after the hexa nucleotides to deca nucleotide and more. The IMEx tool dosent reports some of the repeats which is starting from mono to hexa nucleotide repeats which is reported by the proposed algorithm. Our program reports all the possible repeats from mono nucleotides to deca nucleotide. Using our algorithm we can show repeats up to 25, up to 50, up to 75 and up to 100. For 25 it generates 221 repeats, for 50 it generates 221 and so on. We also run the program on four whole genome sequences Plasmodium falciparum chromosome IV (NC_004318.1), yeast chromosome IV (NC_001136.8), Mycobacterium tuberculosis H37Rv genome (NC_000962.2) and E.coli K12 genome (NC_000913.2). TRF uses the probabilistic algorithm which includes a 'detection step' to identify the candidate repeats and an 'analysis' step that uses different statistical criteria to filter the candidate repeats. Sputnik uses a recursive algorithm and the performance depends on the recursion depth of the program. Hence, Sputnik's execution time seems to be dependent on the sequence composition. IMEx uses the simple string-matching algorithm that scans the entire sequence using sliding window approach and reports the results in a single run. Hence, the processing time of IMEx is dependent on the length of the DNA sequence and not on the number of microsatellites. Our program uses the simple string matching algorithm and report the results in a single run. Hence processing time is not dependent on the DNA sequence file, it depends upon the size of the motif. The program has been designed keeping in view of the limitations we encountered with the other available tools. Using the proposed program the user can search perfect microsatellite and also imperfect microsatellite, generate alignments, set the imperfection percentage threshold of the entire tract of each repeat size, search the repeats of a particular size. From the result, we generate more number of repeats as shown in above Table.1.

Table 2:Comparison of execution times(in seconds) of TRF, Sputnik, IMEx, and proposed program

Sequences	TRF		Suptnik		IMEx		EPI	
	Repeats	Time	Repeats	Time	Repeats	Time	Repeats	Time
Plasmodium Chr4(1204Kb)	25601	69.8	10810	89.1	54232	2.9	**111695**	**5.69**
Yeast Chr4 (1531 Kb)	7308	4.4	2831	287.2	39759	4.0	**54768**	**7.35**
MTB H37Rv (4411Kb)	16439	25.5	9412	17.7	111113	11.6	**131290**	**21.03**
E.coli K12 (4639 Kb)	12043	8.8	5387	8.5	105392	12.3	**129229**	**22.09**

TRF: Match: þ2 Subs: 8 Indel: 8 Min. Score: 20 pM: 0.80 pI: 0.10 Max.Period:6.Sputnik: Match: þ2 Mismatch: 6 Min. Score: 8. IMEx: 'k' value: Mono: 1, Di: 1, Tri: 1, Tetra: 2, Penta: 2, Hexa: 3; 'p' value:10% for all repeat sizes; repeat length: 10 bases or more XXXX: 'k' value: Mono: 1, Di: 1, Tri: 1, Tetra: 2, Penta: 2, Hexa: 3; 'p' value:10% for all repeat sizes; '*n*' value : Mono: 5, Di: 3, Tri: 2, Tetra: 2, Penta: 2, Hexa: 3 repeat length: 10 bases or more.

The above comparison shows that our program reports more number of repeats than the mentioned three tools (TRF, Sputnik, and IMEx). So we can say that our program is more efficient, more accurate and flexible than the other ones.

4. CONCLUSION:

In this paper we have presented a new algorithm for finding perfect and imperfect microsatellite repeats in DNA sequences. In this first we have found all perfect repeats in DNA sequences and then also we have to find imperfect. And then we have stored them in to a text file. for finding tandem repeats is wide ranging and non-standardized. One has to be careful in understanding the tools' inherent constraints to select the right tool for the right purpose. It is hence important to be able to compare the repeat search tools and understand their behavior and inherent limitations.

REFRENCES:

[1] Anwar,T. and Khan ,A.U. (2006) SSRscanner: a program for reporting distribution and exact

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

location of simple sequence repeats Bioinformation,1, 89–91.

[2] Archak,S. et al. (2007) InSatDb: a microsatellite database of fully sequenced insect genomes.Nucleic Acids Res., 35, D36–D39.

[3] Ross, Charles L., et al. "Rapid divergence of microsatellite abundance among species of Drosophila." *Molecular Biology and Evolution* 20.7 (2003): 1143-1157.

[4] Benson,G. (1999) Tandem repeats finder: a program to analyze DNA sequences. Nucleic Acids Res., 27, 573–580.

[5] Bizzaro,J.W. and Marx,K.A. (2003) Poly: a quantitative analysis tool for simple sequence repeat (SSR) tracts in DNA. BMC Bioinformatics, 4, 22.

[6] Boby,T. et al. (2005) TRbase: a database relating tandem repeats to disease genes in the human genome. Bioinformatics, 21, 811–816

[7] Boeva,V. et al. (2006) Short fuzzy tandem repeats in genomic sequences, identification, and possible role in regulation of gene expression. Bioinformatics, 22, 676–684.

[8] Castelo,A. et al. (2002) TROLL – Tandem repeat ocurrence locator. Bioinformatics, 18, 634–636.

[9] Ellegren,H. (2004) Microsatellites: simple sequences with complex evolution. Nat. Rev. Genet., 5, 435–445.

[10] Kashi,Y. and King,D.G. (2006) Simple sequence repeats as advantageous mutators in evolution. Trends Genet., 22, 253–259.

[11] Kolpakov,R. et al. (2003) mreps: efficient and flexible detection of tandem repeats in DNA sequences. Nucleic Acid Res., 31, 3672–3678.

[12] Katti,M.V. et al. (2001) Differential distribution of simple sequence repeats in eukaryotic genome sequences. Mol. Biol. Evol., 18, 1161–1167.

[13] Li,Y.C. et al. (2004) Microsatellites within genes: structure, function, and evolution. Mol. Biol. Evol., 21, 991–1007.

[14] Meloni,R. et al. (1998) A tetranucleotide polymorphic microsatellite, located in the first intron of the tyrosine hydroxylase gene, acts as a transcription regulatory element in vitro. Hum. Mol. Genet., 7, 423–428.

[15] Parisi,V. et al. (2003) STRING: finding tandem repeats in DNA sequences. Bioinformatics, 19, 1733–1738.

[16] Suresh B. et al. (2007) IMEx: Imperfect Microsatellite Extractor Bioinformatics, 1181–1187.

[17] Sturzeneker,R. et al. (1998) Polarity of mutation in tumor-associated microsatellite instability. Hum. Genet., 1 0 2, 231–235.

[18] Tautz,D. and Schlotterer,C. (1994) Simple sequences. Curr. Opin. Genet. Dev., 4, 832–837.

[19] Thibodeau, Stephen N., G. Bren, and D. Schaid. "Microsatellite instability in cancer of the proximal colon." Science 260.5109 (1993): 816-819.

[20] Sreenu VB, et al . MICAS: a fully automated web server for microsatellite extraction and analysis from prokaryote and viral genomic sequences. Appl. Bioinformatics 2003;2:165-168.

[21] Temnykh S, et al . Computational and experimental analysis of microsatellites in rice (Oryza sativa L.): frequency, length variation, transposon associations, and genetic marker potential. Genome Res. 2001;11:1441-1452.

[22] Wexler Y, et al . RECOMB 2004. 2004. Finding approximate tandem repeats in genomic sequences.

[23] AbajianC.Sputnik. http://espressosoftware.com/pages/sputnik.jsp.