MIDAS: Computer application for the identification of exact and inaccurate microsatellites in genomic sequences.

MIDAS: Aplicación informática para la identificación de microsatélites exactos e inexactos en secuencias genómicas.

Carlos M. Martínez Ortiz [1*]

[1]Department of Biochemistry, ICPB "Victoria de Girón", University of Medical Science, La Habana, Cuba

[*]email address: cmmo@infomed.sld.cu

ABSTRACT

Microsatellites are tandem repeat, frequent and diverse short sequences in the genomes of all species, constituting important markers in multiple areas of genomics-based research. Associations of these markers have been found in a significant number of human diseases. Vaccine development has shown how pathogens can evade the immune response by simply altering the composition of repeat sequences in their genes. There are numerous computer applications for the detection of these sequences, but they do not meet all expectations due to the divergence of criteria and approaches applied to solving the problem of their detection. MIDAS implements a non-heuristic solution based on two combinatorial algorithms in series: the first one detects exact microsatellites, and the second one, if the model parameters allow it, extends the sequences to their optimal inaccurate version. The application has as input the genomic sequence in GBFF or FASTA format and its output provides the microsatellite positions in the genomic sequence, as well as sizes, alignments, flanks and other statistics. The algorithm is highly efficient and comprehensive, detecting all possible repeat sequences regardless of their nucleotide composition.

Keywords: SSR; microsatellite; molecular marker; data mining; algorithms

RESUMEN

Los microsatélites son secuencias cortas repetidas en tándem, frecuentes y diversas en los genomas de todas las especies, constituyendo importantes marcadores en múltiples áreas de investigación basadas en la genómica. Se han encontrado asociaciones de estos marcadores a un número importante de enfermedades en humanos. En el desarrollo de vacunas se ha demostrado cómo los patógenos pueden evadir la respuesta inmune simplemente alterando la composición de las secuencias repetidas en sus genes. Existen numerosas aplicaciones informáticas destinadas a la detección de estas secuencias, no obstante éstas no cubren todas las expectativas

debido a la divergencia de criterios y enfoques aplicados a la solución del problema de su detección. MIDAS implementa una solución no heurística basada en dos algoritmos combinatorios en serie: el primero detecta microsatélites exactos, y el segundo, de permitirlo los parámetros del modelo, extiende las secuencias a su versión inexacta óptima. La aplicación tiene como entrada la secuencia genómica en formato GBFF o FASTA y su salida brinda las posiciones de los microsatélites en la secuencia genómica, así como tamaños, alineamientos, flancos, posiciones, etc. El algoritmo tiene una elevada eficiencia y es exhaustivo, detectando todas las posibles secuencias repetidas independientemente de su composición nucleotídica.
Palabras Clave: SSR, microsatélite, marcador molecular, minería de datos, algoritmo.

Introduction

Microsatellites are short tandem repeat sequences (known by their acronyms STR for short tandem repeats or SSR for simple sequence repeat) with repetition units between 1 and 6 bps (some authors extend their definition up to 8 bps), which can be tracts of repetitions ranging from a few copies to hundreds of copies. These sequences are abundant in the eukaryotic genomes, mainly associated with, but not exclusive, to non-coding regions. They are also present in prokaryotes genomes, constituting important markers for the genotyping, classification and epidemiological control of species of interest [(1)]. Among the main motivations for the study of these sequences are their participation in processes such as recombination and transcription regulation [(2)], and when found in coding regions they cause neurodegenerative conditions such as fragile X syndrome, Huntington's disease (HD), spinobulbar-muscular atrophy (SBMA), Haw River syndrome (DRPLA), spinocerebellar ataxias (SCA1, SCA2, SCA3, SCA6, SCA7, and SCA17), as well as some types of cancer [(3,4)]. Vaccine development has shown how pathogens can evade the immune response by simply altering the composition of repeat sequences in their genes [(5)]. It has been proved how microsatellite expansion and contraction in bacteria can regulate expression of specific genes or affect its coding sequence resulting in phase or antigenic variation. This is particularly advantageous in pathogenic bacteria at contingency loci, as a way to evade the defense strategies of its host [(6,7)] . As genetic markers they have been widely used in genetic population studies due to their high polymorphism as a consequence of their high mutation rates, allelic diversity, co-dominance and being selectively neutral. Its use in forensic medicine for the identification of persons and degree of kinship is also well known.

The microsatellites have been experimentally identified from genomic libraries of organisms of interest, inspecting thousands of clones by hybridization with microsatellite probes. In addition to their high cost, these methods contain the bias inherent in the composition of pre-selected sequence patterns. With the modernization and lower cost of sequencing technologies, along with collaborations for the public exchange of sequences such as GenBank, EMBL, DDBJ, among others, bioinformatics methods have taken supremacy, giving rise to numerous applications that implement algorithms oriented to this end. However, the very dynamics of these sequences, subjected to different evolutionary forces, their particular roles, as well as the particular interest of researchers in one or the other depending on their composition, general features and biological purpose, has led these bioinformatics applications to implement different computational criteria, and consequently, to show variations in their results [(8,9)]. To cite a few examples, we find applications that extend their detection system to repeat regions with longer periodicities (i.e. minisatellites and satellites); others detect only exact repeats or with minimal variations defined a priori; others use preset dictionaries of microsatellites or detect regions of low complexity and then confirm those using established rules. Applications such as TRF, IMEx, START, SRF or TROLL [(10, 11, 12, 13, 14)] are examples of software widely used for mining these kind of sequences, implemented with different algorithmic criteria. The conclusion is that there is not a unique solution, the spectrum of applications is diversified according to the types of SSR of interest and the computational methods used, being reflected in result’s differences.

The application we present, MIDAS (MIcrosatellite Detection Assistant System), fulfills with the following general principles: 1st only exact or inaccurate microsatellites are detected (i.e. not composite nor complex tandem repeats with patterns between 1-8 bp); 2nd exact microsatellites are detected and then extended if their flanks show a high sequence similarity with the repetition pattern; 3st the exact detection is exhaustive (i.e. all possible patterns that could make up an SSR are taken into account); and 4th extension is done by local alignment providing an optimal solution according to pre-determined alignment parameters.

The following section, Methods, describes in detail the sequence of steps followed by the application, algorithmic fundamentals, particular solution proposed to the problems of inaccurate detection of SSRs, and examples of detection of exact and inaccurate SSRs. It also describes the parameters and input - output formats of the application.

In the Results section, it is exhibited and analyzed the outputs corresponding to the detection of SSRs for Salmonella enterica (subsp. enterica Serovar Cubana). The input parameters and output formats of the application are described too.

Methods

The procedure starts with the detection of an exact repeat sequence, i.e. without substitutions, insertions or deletions of bases. For this purpose, the Aho-Crasick automaton (ACA) is implemented in the first stage, which finds all the occurrences of words in a text from the construction of a word tree. In the proposed implementation, a tree is constructed that contains all the words, of sizes between 1-8 nucleotides (one of the parameters of the application allows setting the upper limit of this range), formed by combinations of the 4 nucleotide bases, and with the specificity that they do not themselves constitute repeat sequences (e.g. aaaaaa) and excluding its cyclic permutations. ACA computes the search for these occurrences efficiently in time proportional to the size of the text and without pre-processing it. The occurrences of adjacent identical words are spliced and their position recorded, establishing the exact repeats or "seeds" of possible inaccurate repeats. With this step, the algorithm behaves like any other application that detects exact repetitions in an exhaustive and deterministic way (Fig. 1 (I)). The limitations of programs that detect only these sequences are obvious in terms of the biological purpose pursued. Let's think, just as an example, of a repeat that has a simple modification in a base, in which case the program would detect two repeats of the same class separated by a base, when in fact they were part of the same repeat. The generalized version of this problem creates an infinite number of situations that are less trivial and complicated to exemplify, and constitutes the main motivation for the proposed solution. In short, it is a matter of detecting an exact repeat "seed", with a reasonable number of repetitions not occurring by mere chance, from which to detect, if it exist, the inaccurate repeat one of which it is part. If there is no such approximate or inaccurate extension, the corresponding exact repetition will be reported, in this case free of ambiguity.

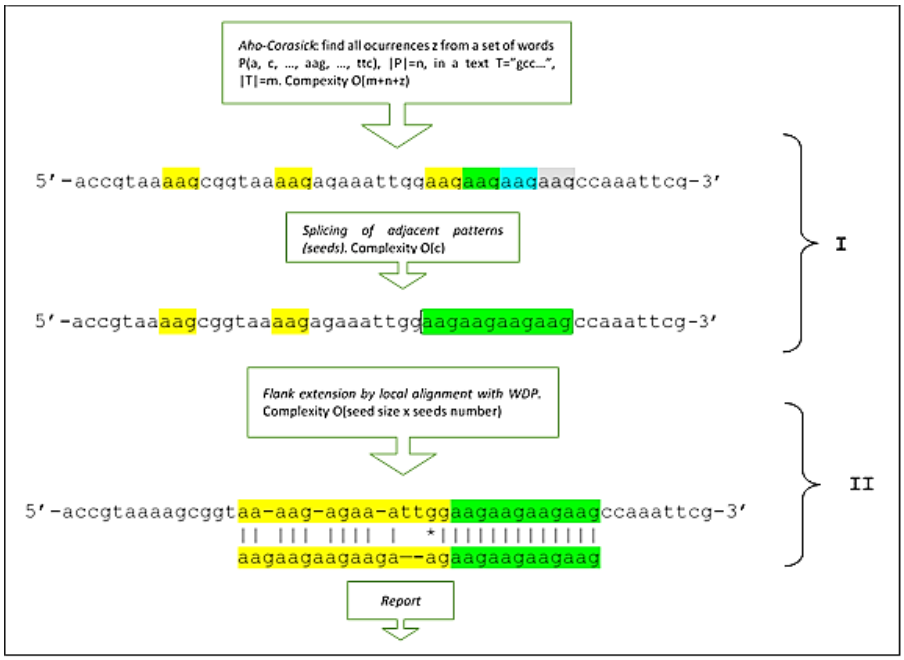

Fig. 1- Sequence of steps of the algorithm implemented in MIDAS in its two fundamental stages. (I) Detection of exact microsatellites (seeds), using word tree (Aho-Corasick) and subsequent splicing of occurrences. (II) Seed extension and detection of possible inaccurate microsatellite using dynamic programming.

The second stage of the algorithm solves the problem described above. The aim is to search for the possible inaccurate candidate from the flanks of the exact seed previously detected. Dynamic programming, i.e. the local alignment of the problem sequence, including flanks, against the repetition pattern, is used at this stage using the efficient wraparound technique (WDP, wraparound dynamic programming) (Fig. 1(II), Fig. 2). As is typical in sequence alignment methods, the optimal solution is dependent on the alignment parameters that define the weightings by coincidence, substitution or insertion/deletion of bases, which will ultimately determine the degree of conservation of the reported microsatellite.

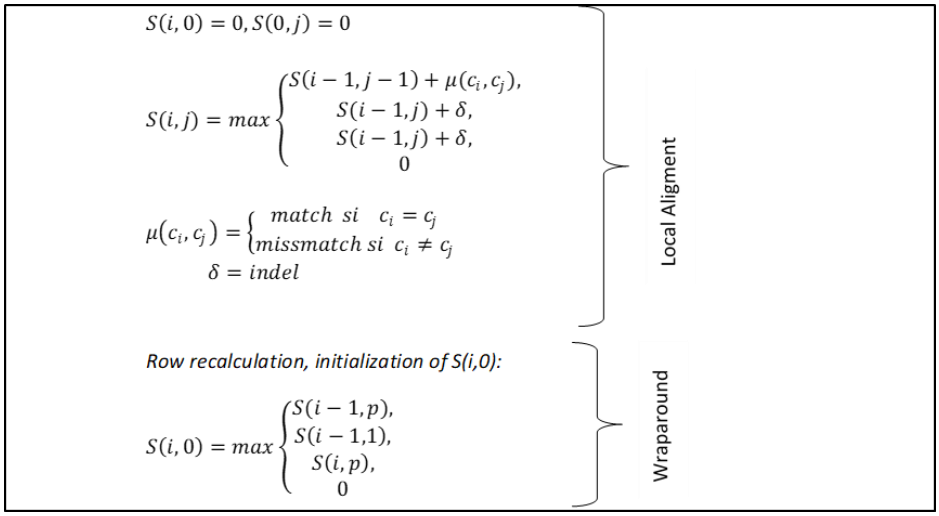

Fig. 2. Recurrence that defines the algorithm of the second stage in MIDAS. It is a classic local alignment applying wraparound technique. Based on the repetition of the pattern, the gain in terms of time and space is established by allowing to align the problem sequence only with the pattern of the microsatellite. Match, mismatch (μ) and indel (δ) are the parameters for matches, substitutions or insertion/deletion respectively.

With respect to the extension per se there are two problems, which appear little explicit in the applications reported by other authors using sequence alignment, and which must be clarified and reasonably solved: 1st, how far do we extend on the flanks of sequence to report the alignment? When the sequence under study is relatively short the problem disappears if we use the all the sequence to find the optimal local repeat subsequence. In most cases this is not possible, think for example of a human chromosome with 200 million of base pairs, and thousands of candidate microsatellites in different regions of the genome. The solution presented by MIDAS is to use flank sizes of 3 times the seed size, and if the computed alignment covers more than 90 percent of the chosen sequence, the flank extension process is repeated and the sequence is realigned. In this way, we guarantee that the region to be extended to look for the inaccurate repeat is dynamic and does not exclude regions where it can continue to be extended. 2nd, How to evaluate if an alignment is adequate to decide to select it? This problem is inherent to all methods of sequence alignment and has been addressed in a variety of ways depending on the context. In applications for tandem repeats, some authors use the alignment score as selection criteria (this is the case of TRF). This approach is somewhat arbitrary, bearing in mind that while the score depends on the parameters of alignment, it also depends on the size of the alignment, and a certain bias is established that favors larger SSRs over shorter ones, being both equally important. In the case of MIDAS this problem disappears taking into account that the starting point is an exact SSR and the extension of the same will fall exclusively on the alignment parameters, in other words the application does not have the need to choose a priori an SSR establishing a cut-off value from the alignment score.

The alignment parameters are by default quite restrictive (Match=2, Mismatch=-5, Indel=-5), although the user has the option of using other more relaxed scoring schemes (4 in total) and then could debug SSRs at will by visual inspection. Figure 3 exemplifies the above with the results of the detection of two SSR in the same genome and with the same scoring scheme.

Fig. 3- Output of the program for two microsatellites located in two positions (separated by 379821 bp) of the genome vibrio cholerae IEC224 (NC_016944). The sequence pattern is the same in both as well as the number of exact repeat units $(aag)_4$. The sequence in green is the exact SSR, in yellow the inaccurate extension and the rest are flanks. The alignment parameters were (Match=2, Mismatch=-3, Indel=-3). In (I) the program extended to a guanine (g) coincident in left flank, however in (II) there is a much greater extension that includes a substitution and five insertions or deletions of bases. Note that this extension would be reduced to an exact repetition if more restrictive parameters were used, (e.g. Match=2, Mismatch=-5, Indel=-5).

Results and Discussion

MIDAS is presented in its version 1.0 (binary) for Windows 32 and 64 bits platform (download v1.0.zip of the supplementary material) and its source code is written entirely in C++ STL compatible, so that it can be compiled for other platforms (Linux, Mac OS, etc.) using the appropriate compilers and libraries. It is a shell application, ideal for batch processing and linking to other applications (pipelines) by command line argument assignment. The application's arguments are three: 1th file name (genome to be scanned in FASTA or GBFF format, both single or multi-locus), 2nd maximum size of the repeat unit to be scanned and 3rd alignment parameter scheme

for match, mismatch and indel, 4 in total, (2,-7,-7) (2,-5,-5,-7) (2,-5,-5) (2,-3,-5). As output, MIDAS returns three text files named equal the input file and with extensions .xls, .dat and .mfaa (the.xls could open directly with Excel or another spreadsheet application). The .xls file (Fig. 5) is in tabular format and its columns are: Pattern, Length (period), Start (initial position in the genome), End (final position in the genome), Score (alignment score), Matches (matching bases), Mismathces (non-matching bases), Indel (insertions and deletions of bases), Inaccuracy (% of inaccuracy of the repeat, measurement of imperfection of the repeat), 5' Flank (flank sequence to the 5’ end), 5' Entropy (compositional entropy in 5´ flank), 3' Flank (flank sequence to the 3´ end), 3' Entropy (compositional entropy in 3´ flank). Some developers of software for tandem repeats detection report the compositional entropy of the repeat region, this being obvious and mostly low. What allows a microsatellite to be used as a genetic marker is the variations in the number of copies and the uniqueness of flanks for its amplification in PCR techniques. MIDAS reports the entropy of the flanks, being these candidates for primer design in PCR technique, and giving a measure of how informative and unique they may be in the genome.

The file with .dat extension presents the previous data in a non-tabular form and allows the sequence alignment to be displayed. Finally, the file with extension .mfaa presents the detected microsatellites in multi-fasta format, with the repeat region marked in lower case and the flanks in upper case. This format allows batch processing with blastn (in Filter and Masking Options, mask lowercase letters checkbox), for the search of intra- and inter-species polymorphic candidates (Fig. 4). The header of this file presents information such as the GenBank access number, the pattern and the positions in the genome.

Fig. 4-File format with extension .mfaa (multi-fasta with repeat region marked with lowercase letter).

The genome of Salmonella enterica (subsp. enterica Serovar Cubana str., access code NC_02181818) taken from the NCBI repository (https://www.ncbi.nlm.nih.gov/), was scanned with MIDAS and the results are shown above (Fig. 5). This genome has 4.977.480 base pairs and the computation time, including the creation of the output files, was less than 3 seconds. A total of 95 SSRs were detected, 2 hexa-, 14 tetra-, 70 tri-, 7 di- and 2 mono-nucleotides (number and repeat unit respectively) The detection parameters were: repeat unit <=6 and Match=2, Mismatch=-3, Indel=-3 (type 4 scheme of alignment parameters for the extension phase). The percentage of SSRs with tri-nucleotides (74%) is notable, which makes it suspicious of their location in coding regions, despite the fact that these regions predominate in bacterial genomes. The

copy numbers show a range of 3 to 15, with an average of 5.8, for a coefficient of variation of 50%, highlighting among these the SSRs numbers 1 and 2 (hexa-nucleotides with 14 copies), 37, 75 and 76 (of tri-nucleotides with 13, 14 and 15 copies respectively) and 93 (of di-nucleotides with 13 copies).

	Accession: NC_021818.1													
No.	Pattern	Length	Copies	Start	End	Score	Matches	Mismatces	Indel	Inaccuracy(%)	5' Flank	5'Entropy	3' Flank	3'Entropy
1	accacg	6	14	4124875	4124962	131	79	9	0	10.2	gcagcagtggctagcgggaa	1.82	tcatggtcacatacatccgg	1.99
2	accatg	6	14	4124875	4124963	133	80	9	0	10.1	gcagcagtggctagcgggaa	1.82	catggtcacatacatccgga	1.97
3	aaac	4	3	4539606	4539617	24	12	0	0	0	agaacctcttatgaaattca	1.85	tcagccttaatcttatgctt	1.82
4	aaat	4	3	1341668	1341679	24	12	0	0	0	tcaaactggtgatatatggg	1.9	gtgagagagttatatttccg	1.88
5	acgc	4	4	1895851	1895869	28	18	1	1	10	cgatctggcgcgtctctttg	1.79	ggcgccaggagagagactta	1.85
6	agcc	4	3	163898	163910	26	13	0	0	0	gcgacgcgtaccgaagcggt	1.85	gcttaccggaattaacatag	1.96
7	agcc	4	4	2430728	2430743	27	15	1	0	6.25	aggccgaaggtgccgagaat	1.85	gttccgcctggtaacgagta	1.99
8	agcc	4	5	3149711	3149730	30	18	2	0	10	taccggtagccccgccgcgg	1.71	tcgcctgacattgtcgcgtt	1.88
9	agcg	4	3	2599506	2599517	24	12	0	0	0	caaagaaacgccgatatgaa	1.76	gcctcgcgccaggacattaa	1.94
10	aggc	4	4	888148	888163	32	16	0	0	0	ttttttccttacgttagtta	1.6	cggccgccgcagagtgccgg	1.6
11	cctg	4	3	1043085	1043096	24	12	0	0	0	cgcttcctgaaaaagcgttc	1.99	gaagagcgacgtttgctggt	1.91
12	cgtt	4	3	2175064	2175075	24	12	0	0	0	gcgaggctaaaaacgcttcg	1.95	ttatcggtcaggcactggct	1.95
13	ctgg	4	4	197857	197875	33	18	1	0	5.26	ggtcggcgcaacggagcaaa	1.77	gatgcgcccggatctgacct	1.93
14	ctgg	4	3	836393	836407	30	15	0	0	0	aatgggcggagcgtcatcgc	1.91	gacctcgctatcaattgtgca	1.99
15	ctgg	4	9	3751746	3751782	29	29	8	1	23.7	gtgctgaccggtatttatat	1.91	gtcgtcgcgcaacaaaccac	1.85
16	ggtt	4	5	218111	218131	27	18	3	0	14.3	taacctttggcaacgctac	1.95	agtgtatggcccgcgctgtt	1.88
17	aat	3	11	3213661	3213695	30	28	7	1	22.2	cctcatgaattcttgggaaa	1.99	caaaagttgcgaaataaata	1.68
18	acc	3	9	480757	480784	31	23	5	0	17.9	gaatcctccatcggtgactg	1.99	taacatttccacaaagacct	1.77
19	acc	3	4	1138726	1138737	24	12	0	0	0	tgaccggcaccaatggcaag	1.9	caactactggcgcagtggag	1.94
20	acc	3	5	4039134	4039149	27	15	1	0	6.25	tgaagctacaaattcagggg	1.93	gcggggagcaggttctgcaa	1.86
21	acc	3	4	4670518	4670530	26	13	0	0	0	cagggtcaagcggcgcgggg	1.6	gcctcgcgtttcgtaaaatt	1.96
22	agc	3	8	237150	237175	35	24	1	2	11.1	cagagcggattgggtctggt	1.84	gatccactattccgcaatgc	1.94
23	agc	3	10	1119681	1119712	39	27	5	0	15.6	ccatcgaccaccacggcgac	1.68	cgattaataaggcgtccgtc	2
24	agc	3	4	1734328	1734340	26	13	0	0	0	gactgacgtttgagatttac	1.94	tctcgctggagggtccgcag	1.85
25	agc	3	4	4460210	4460221	24	12	0	0	0	gcgccattgctcatgaaaat	1.99	cgaatcggcctcctggagaa	1.95
26	atc	3	4	800165	800176	24	12	0	0	0	gcgtctcctcctcgcgacca	1.76	cagcggggagattccgggat	1.86
27	atc	3	5	902834	902850	29	16	1	0	5.88	gttatgatatacttttgca	1.74	agggcatcaacaagtccttc	1.97
28	atc	3	5	1615528	1615544	27	16	0	1	5.88	agcagaataccggacagaat	1.82	ggccgtaaccttcgggcct	1.88
29	atc	3	4	2479307	2479320	28	14	0	0	0	catccgatgtgctgggcagg	1.91	gccgccacagaggatgccga	1.78
30	atc	3	4	3964635	3964648	28	14	0	0	0	tgtagatatcgtcaccggac	2	cccagcgccagtagcacgct	1.82
31	atg	3	6	108398	108416	28	18	1	1	10	cgtcgacgtccgcatcggtc	1.85	aacgcagcttgtccgggttg	1.94
32	atg	3	4	219057	219068	24	12	0	0	0	ggcattccggtgcctgcgtt	1.79	tctttattgttggtgtacat	1.68
33	atg	3	4	1557419	1557430	24	12	0	0	0	tcgcgagccccccgccaac	1.68	gcgatcacccacgcaaatat	1.88
34	ccg	3	4	320384	320397	28	14	0	0	0	gcgcattcaccagtagtcct	1.96	taacggattgccgccgctgg	1.93
35	ccg	3	6	792037	792054	31	17	1	0	5.56	ccattaggcctgccaacgta	1.96	atgccattaataataccgga	1.91
36	ccg	3	4	849993	850006	28	14	0	0	0	ggctttttccacgaccggca	1.94	ttacagataaccaggtgatc	1.96
37	ccg	3	13	917144	917184	31	33	5	4	21.4	caggcggctcagggagactt	1.91	atcagatgaacgtgttggtc	1.95
38	ccg	3	4	1354972	1354985	28	14	0	0	0	ccgcggtggcagcggcgcag	1.58	aggtatggcgcaggatgcgg	1.79
39	ccg	3	4	1510859	1510870	24	12	0	0	0	ggtgatggagcgcctggtcc	1.82	ggcatggctatcgtccggat	1.94
40	ccg	3	6	2201801	2201818	36	18	0	0	0	ggcgaaggccaaatttttta	1.95	tatgaaacgcaaaagcggca	1.82
41	ccg	3	5	3026100	3026116	29	17	0	1	5.56	gaacgcctccaccacctttc	1.76	gtatgcccttccatcaaaat	1.9
42	ccg	3	6	3493256	3493274	33	18	1	0	5.26	cgcagactgagtacgacata	1.94	gaaggtgttcagatcggcaa	1.93
43	ccg	3	5	3745335	3745351	29	16	1	0	5.88	tgagatcgataattcatcaa	1.87	atcgcagcaacgaatgcaga	1.86
44	ccg	3	4	3987068	3987081	28	14	0	0	0	tagatagcggcagcttcacg	1.99	atcggccaaacaaaaagcgg	1.77
45	ccg	3	4	4151511	4151523	26	13	0	0	0	ccgatcggtcccatgccagg	1.87	tgcggaatgcagaaggtttt	1.88
46	ccg	3	5	4370073	4370087	30	15	0	0	0	tcctccgtctgccgttccag	1.72	ggaaacctgctcgaccagcg	1.88
47	ccg	3	9	4465409	4465435	39	24	3	0	11.1	caactgtgccgtcaagttca	1.99	agtgaaattggctcagtaca	1.94
48	ccg	3	7	4591653	4591674	34	20	2	0	9.09	aatactcataaccaaagcga	1.74	gctcgcgtttaccgtcttct	1.74
49	ccg	3	4	4757355	4757367	26	13	0	0	0	gccaacctgccctgggtgtt	1.88	gaggaaccctcctgaagca	1.95
50	ccg	3	5	4818240	4818254	30	15	0	0	0	ctgcgcgctgggttttctga	1.78	ggatgtggtttgccgatgcg	1.78
51	ccg	3	4	4935620	4935632	26	13	0	0	0	tgcactgcgtcatggcgcta	1.95	attttgatacgtccgacgcg	1.99
52	ccg	3	6	4945192	4945211	30	18	2	0	10	cccgtcgcggcgaagcacag	1.74	aaagggcattccgttcaccc	1.96
53	cct	3	9	480759	480785	39	24	3	0	11.1	atcctccatcggtgactgca	1.96	aacatttccacaaagacctg	1.85
54	cgg	3	5	237802	237816	30	15	0	0	0	tcgcgcgccaaccggcgaac	1.72	aaatgaccaaatggtttaat	1.78
55	cgg	3	5	436672	436686	25	14	1	0	6.67	cgctatcggttccagataat	1.99	agagcagtgccatcaggtcg	1.94
56	cgg	3	4	493840	493852	26	13	0	0	0	ttcaaacgtcagacgttcca	1.95	agtgaatttgtagctcagca	1.95
57	cgg	3	4	506644	506656	26	13	0	0	0	gataggaatagcagaaagga	1.58	aaagtcgccccacaaatagt	1.87
58	cgg	3	6	778415	778432	29	17	0	1	5.56	tgaccctcttcacggatgac	1.96	atagtcagcgcgcattgcgg	1.96
59	cgg	3	11	1399278	1399310	31	27	6	1	20.6	aagaacgcatttgatgagcc	1.96	atctggtcgatgttcggtca	1.93
60	cgg	3	4	1420189	1420200	24	12	0	0	0	gtcgcgccaagccctggaac	1.85	gaagatgccaataaacccgt	1.91
61	cgg	3	5	1543714	1543729	27	15	1	0	6.25	cggtgataatgctgtggatt	1.86	ttagcggtccgctgctgatc	1.9
62	cgg	3	6	1598682	1598701	25	17	3	0	15	tcggcgatggccgggattgt	1.82	acctggcgattcgctatggc	1.95
63	cgg	3	10	1698496	1698527	34	27	5	1	18.2	gcgtatgacgtactgattgt	1.93	tctactcggcgcgcaaaggc	1.93
64	cgg	3	5	1828708	1828722	25	14	1	0	6.67	gagcatatagaggcttctgc	1.99	ttccgctatcgacgcggtga	1.95

Fig. 5- Results of the detection of SSRs in the genome of Salmonella enterica (subsp. entericaSerovar Cubana str., access code NC_021818). This representation is the one shown in the output file with .xls extension.

65	cgg	3	4	1898266	1898278	26	13	0	0	0	agcggcttgcgcctgtctga	1.88	ctgggctatctcttcatcgc	1.86
66	cgg	3	4	2477755	2477766	24	12	0	0	0	tccgatcagccgaaaccgct	1.91	aatgcgttacgcctgcggcg	1.93
67	cgg	3	4	2888933	2888945	26	13	0	0	0	tgttgtttcagcaaatcttc	1.86	aattgtaaataatagccctg	1.87
68	cgg	3	4	3883088	3883099	24	12	0	0	0	attctggttgtgccgtcata	1.91	gatggccggtgcggtgccgc	1.65
69	cgg	3	4	3929160	3929172	26	13	0	0	0	cgaaaacagaaacgccagaa	1.44	aaaccacgcgacgccgctag	1.77
70	cgg	3	4	4727117	4727130	28	14	0	0	0	gccatgatgctgctgatcat	1.99	cattcaagcaggtgctggtc	1.99
71	cgg	3	4	4786486	4786497	24	12	0	0	0	cgggaagccgagcaggaaac	1.56	agaccagtcccgccagcggc	1.72
72	cgt	3	4	749920	749931	24	12	0	0	0	aatacggcgccactaccggc	1.86	accggctggctggacaccgt	1.88
73	cgt	3	5	3347418	3347432	25	14	1	0	6.67	gacttaaacaatccgcccag	1.88	ggcgctggttgcatcacgaa	1.96
74	cgt	3	5	3602816	3602832	29	17	0	1	5.56	ccgaacagtttatcgataaa	1.91	catcaccggaaaaccctata	1.8
75	ctg	3	14	360753	360794	54	36	6	0	14.3	acaggctgcgtttgagccca	1.97	tagcgtttgctggcgttgtt	1.68
76	ctg	3	15	424437	424482	42	36	10	0	21.7	aagaaaaaattcggtgtttcc	1.93	aagaaaaaactgaattcgac	1.71
77	ctg	3	4	1101330	1101342	26	13	0	0	0	tgatcccgacggtattcgag	1.99	gtagcgacgctatccagacg	1.95
78	ctg	3	4	1391866	1391877	24	12	0	0	0	tcctggcaggcgcggataaa	1.94	accgatcagcaaggattatt	1.96
79	ctg	3	10	1845945	1845976	34	26	6	0	18.8	tggcgcaggccaggccatca	1.86	gcgattgcgaaaaagttcga	1.93
80	ctg	3	6	1965950	1965968	28	17	2	0	10.5	atgccggatgtgattaccgg	1.96	tttgtcgcgctcggtcatgc	1.79
81	ctg	3	4	2822605	2822617	26	13	0	0	0	agcgtcgtgaagaagaaagc	1.8	aagtggaagaacgcactcgt	1.93
82	ctg	3	4	3247298	3247309	24	12	0	0	0	tactggcgatgatcatgcgc	1.99	gcgtcaatctcgtggctggt	1.88
83	ctt	3	5	226114	226129	27	15	1	0	6.25	gaagacggactgcatatcca	1.94	gaagatgcggaagatcatgc	1.88
84	ggt	3	4	198817	198830	28	14	0	0	0	atccatgaaacggcagacgc	1.88	ataaactcacctatgcgggc	1.97
85	ggt	3	4	3696472	3696483	24	12	0	0	0	gttgccacggcagggtcacc	1.88	tggcgtgattttgataccga	1.93
86	ggt	3	4	3742603	3742615	26	13	0	0	0	ggacccgccagggttttgtg	1.86	acaacgtgagcgtgcggaac	1.88
87	cg	2	6	1227443	1227455	26	13	0	0	0	ggcgcgccaggcgataattt	1.96	tcgtcgggtaagtcaatcgc	1.99
88	cg	2	6	1532556	1532568	26	13	0	0	0	caggcggcgctgaccgtggt	1.78	gaaaaactgggtattaatcc	1.91
89	cg	2	6	2255344	2255355	24	12	0	0	0	caactggcagacgcttatgc	1.99	aatcgacgcccggcagctat	1.94
90	cg	2	6	3019348	3019359	24	12	0	0	0	gacgaagatgaagcgtttgc	1.93	taactacgtcgtcaaattgc	1.95
91	cg	2	8	4109832	4109847	27	16	0	1	5.88	acccgctatcttacgcctgt	1.87	ttccggcatcggtatttgcg	1.88
92	cg	2	6	4111111	4111123	26	13	0	0	0	agctatcaccctaacgccaa	1.8	tggcacagcaggcaacggaa	1.78
93	cg	2	13	4540368	4540393	35	24	1	2	11.1	tgaagatatcagtctgctgc	1.99	gtatcggctatttgccgcag	1.95
94	a	1	11	3004616	3004626	22	11	0	0	0	agcgtcgggttttcttttc	1.68	tccattaaatacaaagtgtt	1.8
95	t	1	10	170557	170566	20	10	0	0	0	tccttagcatctgctaagga	1.99	gcctaaaattacctgattat	1.86

Fig. 5- (cont.) Results of the detection of SSRs in the genome of Salmonella enterica (subsp. entericaSerovar Cubana str., access code NC_021818). This representation is the one shown in the output file with .xls extension.

The average compositional entropy of the 5´ and 3´ flanks is 1.86 and 1.88 respectively, which can be considered high due to their proximity to 2 (maximum value). Among these, the most outstanding are 3´ flank of SSR No. 23 and 5´ flank of SSR No. 30, both with maximum compositional entropy.

Conclusions

In this paper we present MIDAS, an application for detection of accurate and inaccurate microsatellites (SSRs). The algorithm is fully combinatorial and has two general stages or procedures: 1st detection of exact SSRs by the technique of exact text patterns recognition and 2nd extension of them by means of dynamic programming technique. The result, and a brief analysis, of these sequences in the genome of Salmonella enterica (subsp. enterica Serovar Cubana) are shown. The application is efficient and intuitive, featuring low runtimes (processing 4,977,480 bp in 3 sec.) and a minimum number of input parameters which makes it more users friendly. It presents descriptive, tabulated and bioinformatic output formats that allow an easy and very complete visualization for the analysis of the results, also allowing the linking of these with other applications, for example extraction of annotated features in GenBank or detection of polymorphisms through extensive BLAST database searches.

References

1. Liang S, Watanabe H, Terajima J, Li C, Liao J, Tung S, Chiou C. Multilocus Variable-Number Tandem-Repeat Analysis for Molecular Typing of Shigella sonnei. JOURNAL OF CLINICAL MICROBIOLOGY Nov 2007;45(11):3574–80.
2. Martin P, Makepeace K, Hill S, Hood D, Moxon E. Microsat instability regulates transcription factor binding and gene expression. Proc Natl Acad Sci USA. 2005; 102(10):3800-4.
3. Mitas M. Trinucleotide repeats associated with human disease. Nucleic Acids Res. 1997;25(12):2245-54.
4. Arzimanoglou I, Gilbert F, Barber H. Microsatellite instability in human solid tumors. Cancer 1998;82(10):1808-20.
5. Moxon ER, Rainey PB, Nowak MA, Lenski RE. Adaptive evolution of highly mutable loci in pathogenic bacteria. Current Biology. 1994;4:24-33.
6. Moxon R, Bayliss C, Hood D. Bacterial contingency loci: the role of simple sequence DNA repeats in bacterial adaptation. Annu. Rev.Genet. 2006;40:307–333.
7. Bayliss CD, Field D, Moxon ER. The simple sequence contingency loci of Haemophilus influenzae and Neisseria meningitidis. J. Clin. Invest. 2007;107:657–662.
8. Grover A, Aishwarya V, Sharma PC. Searching microsatellites in DNA sequences: approaches used and tools developed. Physiol Mol Biol Plants 2012 Jan–Mar; 18(1):11–19.
9. Leclercq S, Rivals E, Jarne P. Detecting microsatellites within genomes: significant variation among algorithms. BMC Bioinformatics. 2007;8:125.
10. Benson G. Tandem repeats finder: a program to analyze DNA sequences. Nucleic Acids Res. 1999;27:573–580.
11. Mudunuri SB, Nagarajaram HA IMEx: Imperfect Microsatellite Extractor. Bioinformatics. 2007;23:1181–1187.
12. Merkel A, Gemmell N. Detecting short tandem repeats from genome data: opening the software black box. Brief Bioinform. 2008;9:355–366.
13. Zhou H, Du L, Yan H. Detection of tandem repeats in DNA sequences based on parametric spectral estimation. IEEE Trans Inf Technol Biomed. 2009;13:747–755.
14. Castelo AT, Martins W, Gao GR. TROLL: Tandem repeats occurrence locator. Bioinformatics. 2002; 18:634–636.