BIOINFORMATICS ORIGINAL PAPER

Vol. 21 no. 5 2005, pages 582–588
doi:10.1093/bioinformatics/bti039

Sequence analysis

RAP: a new computer program for *de novo* identification of repeated sequences in whole genomes

Davide Campagna, Chiara Romualdi, Nicola Vitulo,
Micky Del Favero, Matej Lexa, Nicola Cannata and Giorgio Valle*

CRIBI, Università degli Studi di Padova, via Ugo Bassi 58b, I-35121 Padova, Italy

Received on May 4, 2004; revised on August 2, 2004; accepted on August 23, 2004
Advance Access publication September 16, 2004

ABSTRACT

Motivation: DNA repeats are a common feature of most genomic sequences. Their *de novo* identification is still difficult despite being a crucial step in genomic analysis and oligonucleotides design. Several efficient algorithms based on word counting are available, but too short words decrease specificity while long words decrease sensitivity, particularly in degenerated repeats.

Results: The Repeat Analysis Program (RAP) is based on a new word-counting algorithm optimized for high resolution repeat identification using gapped words. Many different overlapping gapped words can be counted at the same genomic position, thus producing a better signal than the single ungapped word. This results in better specificity both in terms of low-frequency detection, being able to identify sequences repeated only once, and highly divergent detection, producing a generally high score in most intron sequences.

Availability: The program is freely available for non-profit organizations, upon request to the authors.

Contact: giorgio.valle@unipd.it

Supplementary information: The program has been tested on the *Caenorhabditis elegans* genome using word lengths of 12, 14 and 16 bases. The full analysis has been implemented in the UCSC Genome Browser and is accessible at http://genome.cribi.unipd.it.

INTRODUCTION

Repeated sequences are a common feature of most prokaryotic and eukaryotic genomes and their identification is an essential step for genome analysis and annotation.

The analysis of repeats can be essentially divided into three main tasks: (1) *de novo* identification; (2) clustering similar or associated repeats; and (3) annotation and database management. The first major advance in this field came with Repbase (Jurka *et al*., 1992), a database of repeats that led to the development of computer programs such as RepeatMasker (A.F.A.Smit and P.Green, unpublished data, http://repeatmasker.genome.washington.edu), Censor (Jurka *et al*., 1996) and MaskerAid (Bedell *et al*., 2000) that are still widely used to identify and mask repeats in genomic sequences, by using the data available from Repbase.

Many repeats tend to be little conserved during evolution, thus making the identification of homologous sequences difficult, both within the same genome and between genomes of different organisms. Therefore, as more and more genomic sequences are identified, the necessity for new tools that are able to detect repeats is becoming crucial. This paper will focus specifically on *de novo* identification, while the problem of clustering and classification of repeats has recently been addressed by others (Volfovsky *et al*., 2001; Bao and Eddy, 2002).

So far, two main strategies have been used to address the problem of *de novo* repeat identification: similarity search and word counting. The program RECON (Bao and Eddy, 2002) uses WU-BLAST (Altschul *et al*., 1990, W.Gish, unpublished data, http://blast.wustl.edu) for the initial detection of repeats. However, this and other similarity search programs were designed to perform pairwise alignment and are not very practical for *de novo* repeat identification, in particular for an all-against-all comparison of large multimegabase genomes.

Word counting is an alternative way to approach the problem. The rationale is that a genomic region containing a high number of frequent words is most likely a repeat. Since short words are not very significant, these approaches tend to consider long words. But direct word indexing is impracticable for word lengths above 15–16 bases; therefore, more sophisticated algorithms and data structures are used. Suffix arrays and optimized suffix trees (McCreight, 1976; Manber and Myers, 1993; Kurtz, 1999) are particularly useful for this purpose and have been recently implemented in programs for repeat analysis, such as REPuter (Kurts *et al*., 2001) and FORRepeats (Lefebvre, 2002). One of the main drawback of these methods is the high memory requirement; however, this has been partially overcome by the application of an elegant compression strategy based on the Burrows–Wheeler transform (Burrows and Wheeler, 1994) that allows the analysis of the entire human genome (>3 Gb) by counting its constituent words (Healy *et al*., 2003).

The word length is a very critical parameter in these studies: short words are found too frequently and are not indicative *per se* of a significant repeat. On the other hand, long words (for instance 30 bases words in a 1 Gb genome) are quite significant, but they are unsuitable to detect degenerated repeats. It has been noted that 'several regions annotated as repeats by the UCSC browser have very low word counts, even with 15mer. This is not unexpected as these methods are based on exact matches and some repeats are very ancient and highly diverged' (Healy *et al*., 2003).

Therefore, the main issue is to find the best compromise between long words (resulting in high specificity and low sensitivity) and short words (resulting in low specificity and high sensitivity). This is a very

*To whom correspondence should be addressed.

difficult task because we would like to guarantee both specificity and sensitivity, but methods based on exact word counting cannot improve one aspect without deteriorating the other.

The Repeat Analysis Program (RAP) that we describe in this paper is based on a new algorithm optimized for high resolution repeat analysis. The program works on direct indexing of gapped words, thus allowing the identification of exact as well as inexact patterns. Since many different overlapping gapped words can be produced and counted at any genomic position, the specificity of the signal is considerably improved. Furthermore, the indexing algorithm has been designed to count double-stranded DNA words, using the same index for a word and for its inverse complementary sequence.

SYSTEM AND METHODS

The general strategy that we explored in this work is based on counting the occurrences of all the words of a given length in a genomic sequence; the final aim being the *de novo* identification of repeated sequences, with a sensitivity that permits to find regions duplicated as little as one time. Since there are four possible bases, the number of different words of a given length w will be 4^w. If we set a counter for each possible word we can count all the occurrences very rapidly, in a time proportional to the length of the genome, using 4^w counters. Even if we use single-byte counters, relatively short words of 16 bases would require 4 GB of addressable memory (RAM), that is, the upper limit for 32 bits computers; while a word length of 20 bases would require a terabyte of RAM, which is seldom available even in the most powerful computers.

Given the above problems together with the objective of developing a strategy that can be applied to gigabase-long genomes, this may seem a worthless approach. In fact, we should expect that many words of 16 bases will be found by chance in multiple copies within the genome.

For instance, in a random DNA sequence with equal base composition, 3 billion bases long (i.e. the typical length of one strand of the mammalian genome), each different 16 base word will be found on average 0.7 times; this is because there are 4^{16} (i.e. ∼4.3 billion) different words of 16 bases. Therefore, according to the Poisson distribution we should expect that each word has a probability of $1 - e^{-0.7}$ (i.e. ∼50%) to be present at least once.

Similarly, according to the Poisson distribution, it can be calculated that more than 2000 different 16 base words are expected to be present at a multiplicity of 8 or above. Therefore, considering such a high background, if a word should occur once more because there is a repeat, this would not produce a distinguishable signal. The statistical methods described in this paper are based on a new strategy that allows estimation of the repetitiveness of any genomic region, overcoming the above problem.

The basic strategy of RAP

Figure 1 illustrates the basis of the strategy described in this paper, which allows the identification of repeats in a genomic sequence n bases long, using words where $w \approx \log_4(n)$. That would be 15–16 bases for mammalian genomes. If $w = \log_4(n)$ then each word is found on average a single time and the probability that a specific word is found at least once is ∼63%, independently of the length of the genome, according to the Poisson distribution. After the genomic sequence is read and the word counters updated, an image of the genome is created where at each position the value of the counter of the word is reported that occurs at that point in the sequence (Fig. 1A and C).

As expected, no appreciable increase in counts can be detected in correspondence to the repeated region (Fig. 1A and C, around position 2000); in fact, the repeated region contains words repeated up to six times, while other regions have words repeated even seven or eight times. The signal improves if we consider a window of 50 bases (Fig. 1B and D), but still remains quite low.

It should be noticed that panel A in Figure 1 is very similar to panel C (and so are the related panels B and D), despite the fact that in the first case there is a 100 base repeat in a 64 kb (4^8 bases) sequence, while in the second case the 100 base repeat is in a 1 Mb (4^{10} bases) sequence. In both cases $w = \log 4(n)$ and therefore the chance to find a repeated word is the same, thus producing a very similar figure.

It can be seen from Figure 1B and D that the signal is quite low when $w \approx \log_4(n)$, even if we use a 50 base wide window. A possible solution to improve the signal could be the increase in the window size, but this would compromise the resolution that for many purposes (for instance oligo design) should be as sharp as possible. To overcome this problem we also considered gapped words that allow multiple independent reading on the same position, thus increasing the signal over the background without further extending the window size.

Differently from ungapped words that are made up by a stretch of contiguous bases, gapped words are assembled from the bases found at the positions defined by a given pattern. For instance, the binary pattern 1011001101 will consider the position 1, 3, 4, 7, 8 and 10, producing a different word from those found at the same position using other patterns (Valle, 1993). If a sequence is an exact repeat, then all the gapped words produced by the different patterns will be over-represented and contribute to improve the signal over the background.

Figure 1E and F shows the analysis of the same random sequence of Figure 1A and B, considering 120 different patterns generated with 8 defined bases and 3 undefined bases; it can be seen that the signal becomes clearly visible, particularly in Figure 1F, with a 50 base window. The RAP index is calculated from the sum of occurrences of gapped and ungapped patterns, as further described in the Algorithm section.

Given the above conjecture, the approach based on counting $\log_4(n)$ long words has several interesting potentials: (1) it is sensitive enough to detect single repeats; (2) the combination of short word statistics and gapped patterns allows us to also see inexact repeats; (3) the signal will increase with the number of occurrences, with the level of similarity and by optimizing the window size in function of the length of the repeat; and (4) both execution time and memory requirement are roughly proportional to the genome length.

Computer system and availability

For the development of the algorithm and the implementation of RAP we used a 2 GHz Athlon bi-processor computer with 4 GB RAM, running under the Linux operating system. On such a system we could perform all the *Caenorhabditis elegans* analysis described in this paper and, using the same system, we are currently extending our analysis to mammalian genomes. More recently, we have successfully optimized and implemented our algorithm on a 64 bits Appro 110H-A1 1U Dual Opteron 146 workstation, equipped with 8 GB RAM. The speed of elaboration on the 64 bits machine is about 150 Mb/min for each considered pattern (see Algorithm section). All the computer programs were written in the C language and are available upon request to academic users. The results of the analysis can be accessed at our UCSC Genome Browser mirror site (http://genome.cribi.unipd.it).

ALGORITHM

A data structure for counting dsDNA words

Genomic DNA is double-stranded (dsDNA). For any subsequence on one strand there is an inverse-complementary sequence (hereafter referred to as inv/comp) on the other strand. For the purpose of the RAP program, it would not make sense to count a word and its inv/comp as two separate items, being actually the same dsDNA word. In fact, it would be detrimental for both memory requirement and execution time.

RAP organizes the word counters as a multi-array data structure (dsDNA data structure) optimized for pointing to dsDNA words. The current version of RAP only works with word containing an even number of bases. The number of required word counters is $(4^w + 2^w)/2$, because each pair of inv/comp words uses a single counter, with the exception of double strand palindromes, where

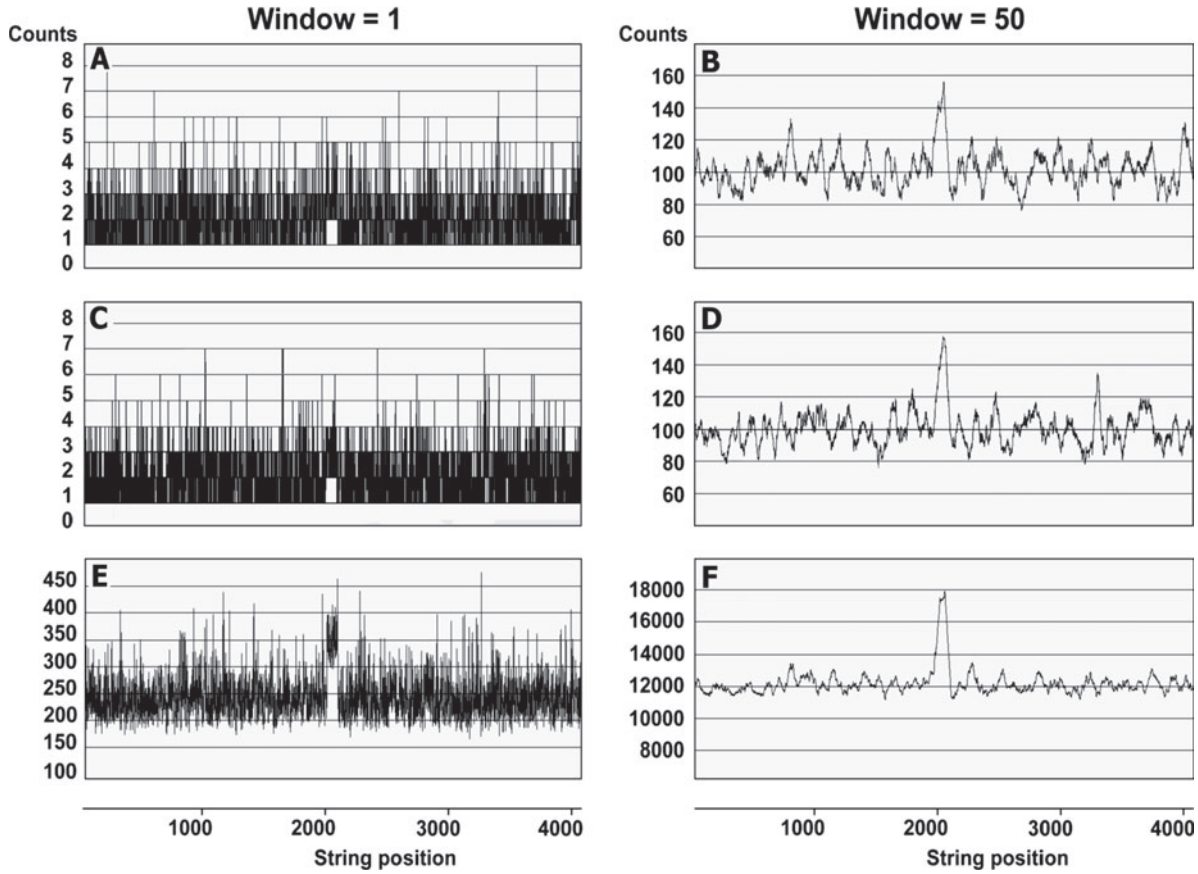

Fig. 1. Basic strategy used by RAP. (**A**) Analysis of a random DNA sequence of 65 536 bases, with an exact repeat of 100 bases at position 2000. After counting the occurrences of all the 8-base words, the value of the corresponding word counters are reported at each position. Only the first 4000 bases are shown in all six panels. (**B**) As in (A), but each position shows the sum of a 50 bases window. (**C** and **D**) Analysis of a random sequence of 1 Mb, with an exact repeat of 100 bases at position 2000 and word length of 10 bases. The results are very similar to A and B since in both cases $w = \log_4(n)$ and therefore the chance to find a repeated word is the same. (**E**) Data obtained from the sum of 120 gapped patterns, as described in the text. (**F**) As in (E), but each position shows the sum of a 50 bases window. The signal corresponding to the repeat becomes very evident and is used for the RAP index (see text).

word and inv/comp coincide. Each word has an associated index that is calculated as follows:

$$\text{index} = b_{w-1}4^{w-1} + b_{w-2}4^{w-2} + \cdots + b_1 4^1 + b_0 4^0,$$

where b indicates the value of the base that is 0, 1, 2 and 3, respectively for A, C, G and T, starting from position 0 (rightmost) to position $w - 1$.

At every genomic position RAP calculates the indexes of both word and inv/comp, then it takes only the index with the lower value and increments the corresponding counter. However, the actual indexing method is more complex because each index is divided into two parts, each related to half word, as described below.

Figure 2 displays a schematic diagram showing the indexes used by the 256 words of 4 bases. Each word is divided into two half-word 'nibbles'. The dots indicate the words whose indexes are lower than their inv/comp, or equal in the case of the palindromic words. Therefore, if a word is without dot then its inv/comp should be used instead. It is relevant to notice that all the palindromic words correspond to the terminal dot of each column.

		Second nibble															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
First nibble	0 AA	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
	1 AC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
	2 AG	●	●	●	●	●	●	●	●	●	●	●		●	●	●	
	3 AT	●	●	●	●	●	●	●		●	●	●		●	●	●	
	4 CA	●	●	●		●	●	●		●	●	●		●	●	●	
	5 CC	●	●	●		●	●	●		●	●	●		●	●		
	6 CG	●	●	●		●	●	●		●	●			●	●		
	7 CT	●	●	●		●	●			●	●			●	●		
	8 GA	●	●			●	●			●	●			●	●		
	9 GC	●	●			●	●			●	●			●			
	10 GG	●	●			●	●			●				●			
	11 GT	●	●			●				●				●			
	12 TA	●				●				●				●			
	13 TC	●				●				●							
	14 TG	●				●											
	15 TT	●															

Fig. 2. Schematic diagram of the 256 words of 4 bases, divided into two half-word 'nibbles'. The dots indicate that the index of one word is smaller than its inv/comp, or equal in the case of the palindromic words.

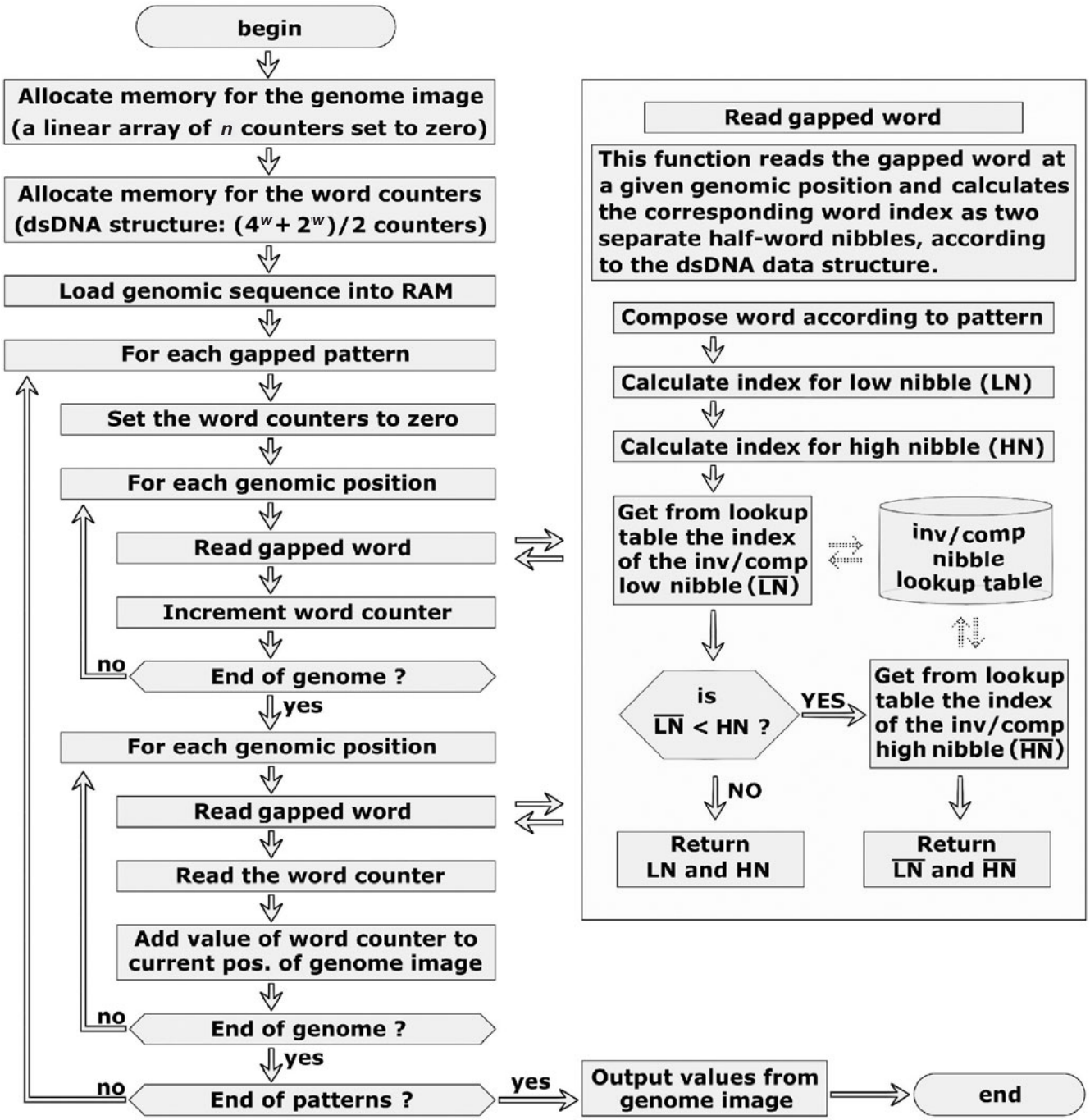

Fig. 3. Flow chart showing the main steps of the RAP program. The length of the genome is indicated by n, while the word size by w.

A number of 2^w arrays is set before the counting starts. Each array contains the counters corresponding to the words ending with the same nibble (the second nibble); therefore, in the example of Figure 2 each array corresponds to one column of dots. In Figure 2 it can be seen that each column of dots has a different size, ranging from 1 to 2^w. The dots are always starting from position zero and continue uninterruptedly until the last dot. Therefore, the size of each array can be easily calculated from the inv/comp nibble because, as mentioned above, each column ends with a palindromic sequence and therefore the last element will correspond to the inv/comp of the second nibble. For instance, the array number 2 will count all the words ending with 'AG'; the last element corresponds to 'CT' row (the inv/comp of 'AG') whose index is 7. Therefore, considering that the numbering starts from zero, the array 2 will be set to contain 8 counters. The same procedure can be applied to any even-length word, while it would require a slight adaptation for odd-length words

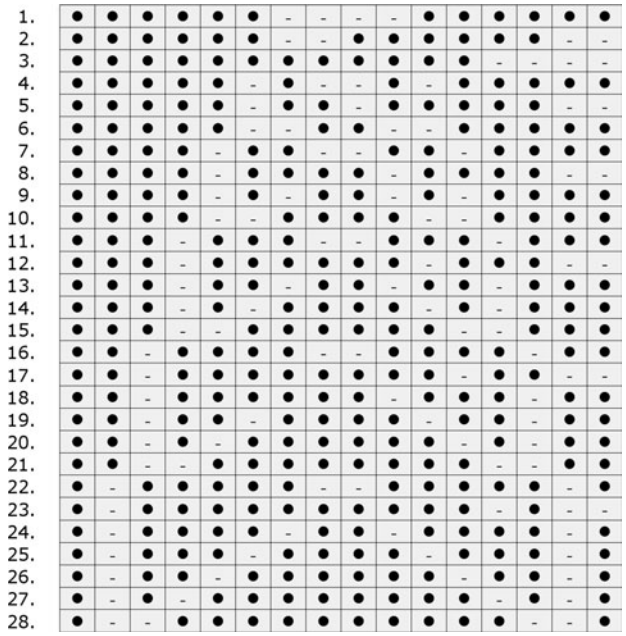

Fig. 4. The 28 possible symmetric patterns generated by 12 defined bases (dots) and 4 undefined bases. A gapped word of 12 defined bases is read from the genome using the 12 positions indicated by the dots.

in which palindromes are not possible and the number of required indexes is $(4^w)/2$.

As mentioned above, at every genomic position RAP calculates the index of the word starting at that position, considering the bases defined in the current gapped pattern (see next section). The index is calculated as two half-word nibbles, as shown in the right panel of Figure 3. To determine whether the actual word or its inv/comp has a lower index, RAP uses a lookup table. Although in principle it would be possible to set up a lookup table for each of the 4^w words, this would require more memory than the word counters. Moreover, the dsDNA data structure works with the two half-word indexes, therefore, instead of a full-word lookup table RAP uses a nibble lookup table, whose size is only 2^w. As shown in Figure 3, it is enough to compare the high nibble with the inv/comp of the low nibble to decide whether the actual word or its inv/comp should be used.

Once the high and low nibbles have been calculated, the corresponding counter is incremented: the low nibble defines the array, while the high nibble defines the position within the array.

This data structure has several advantages: (1) both word and inv/comp are considered as the same item; (2) indexes can be calculated extremely rapidly during the time-consuming genomic scan, without requirements for additional arithmetic operations; (3) there is no waste of memory as all the memory allocated is actually used for the counters and (4) most operating systems require the memory of an array to be contiguous, therefore having the counters divided into many small arrays, allows a better allocation of the available memory.

Generation of symmetric gapped patterns

The potential of using gapped pattern to increase the sensitivity of the signal is one of the main features of RAP and has already been mentioned in the previous section describing the basic strategy of the program. Unfortunately, gapped patterns and inv/comp words are not easily compatible because the complementarity must also include the pattern of gaps that must remain the same on both strands. To overcome this problem we use only symmetrical gapped patterns, as shown in Figure 4 where each line represents a pattern with 12 defined bases (shown as dots) and up to 4 undefined bases.

The gapped patterns must satisfy two main conditions: (1) they must be symmetric in the region included between the first and the last defined base and (2) the first defined base must always occupy the first position to avoid counting the same pattern twice. The symmetric patterns are obtained by a simple algorithm that generates all the possible half patterns and produces symmetric full patterns by joining a half pattern to its mirror image.

Even if only symmetric patterns are considered by RAP, the number of possibilities is quite high. For instance, with 16 defined bases, the number of possible symmetric patterns is 8, 44, 164, 494, 1286 and 3002, respectively for 2, 4, 6, 8, 10 and 12 undefined bases.

Calculation of the RAP index

The RAP index is calculated for each position of the genome, essentially as indicated in Figure 3. By default, the actual value of the RAP index is normalized by dividing the observed counts by the average value expected in a random DNA sequence.

Alternative modes have also been investigated, giving interesting results; for instance, the possibility that only the highest scoring gapped word is considered at every genome position. Some of these alternative modes have been implemented in the current release of the program and can be selected at the command line.

IMPLEMENTATION

The RAP program is written in C language. It takes as input the genomic sequence as well as several parameters defining the word length and the type of analysis. By defaults, it calculates the RAPindex as described in the previous sections. Alternative modes of analysis are also possible, including the possibility of recording the highest or lowest counts rather than the sum of the counters. As an output, RAP produces a binary file containing the value of each genomic base.

The bintosql script is used to read the binary file produced by RAP and to create SQL tables that can be directly integrated into the UCSC genome browser (Kent *et al*., 2002).

RESULTS AND DISCUSSION

As mentioned in the introduction, repeat analysis algorithms based on word counting are very much affected by the word size. Here, we show that the strategy of generating different gapped words at the same genomic position is very effective in improving the quality of the signal. Short words can actually produce better results than long words because they can improve the detection of degenerated sequences in which long stretches of exact matches can be very rare.

A comparison of the results obtained with different word lengths is shown in Figure 5 where words of 12, 14 and 16 defined bases were used to calculate the RAP index at every genomic position of *C.elegans*. Each profile was obtained adding up the results from 200 gapped patterns at every position.

The top panel of Figure 5 shows a typical region of the *C.elegans* genome. All repeated regions identified by RepeatMasker have evident signals in the RAP12 and RAP14 tracks, while the RAP16 signal appears lower, but still indicative of most repeats. It can also be

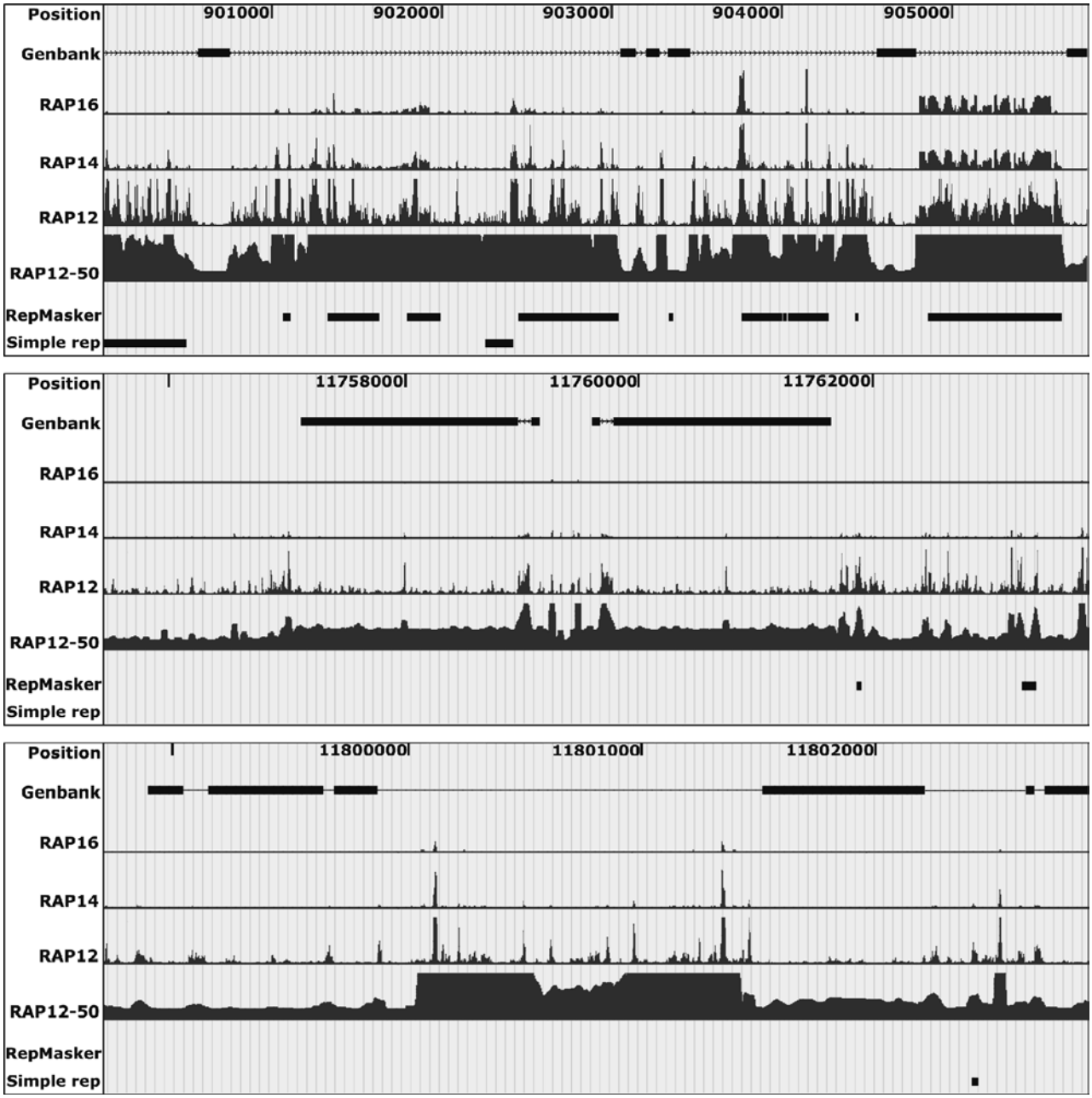

Fig. 5. Analysis of the RAP index on three regions of chromosome I (top panel) and chromosome II (central and bottom panels) of *C.elegans*, as shown by the UCSC genome browser. The numbers at the top of each panel indicate the position on the chromosome; the GenBank tracks indicate predicted genes. The RAP 16, 14 and 12 tracks show the actual results obtained with words of 16, 14 and 12 bases, while RAP12–50 shows the average score in a 50 base long window. For these analysis 200 gapped patterns were considered. At the bottom of each panel are reported the repeats found by RepeatMasker and by TandemRepeatFinder, as annotated on the UCSC genome browser. Further details are given in the text.

appreciated that different parts of each repeat can be more conserved than others, resulting in peaks of the RAP indexes. It must be noted that corresponding to the five exons (thick boxes in the GenBank track) the RAP signals become practically absent. This is better appreciated in the RAP12–50 track where the average values of RAP12 are calculated in a window of 50 bases.

Although the *C.elegans* genome has been extensively studied and annotated, it can be seen that some repeated regions are not yet reported. Two examples of such repeated regions are shown in the central and bottom panels of Figure 5. In the central panel is clearly evident a region containing two genes with a similar structure, going in the opposite direction. A closer analysis revealed that the two genes are identical, encoding a protein containing the Hsp70 motif.

An interesting argument is how much the repeats found by RAP correspond to those detected by RepeatMasker, available in the *C.elegans* genome browser. It should be considered that

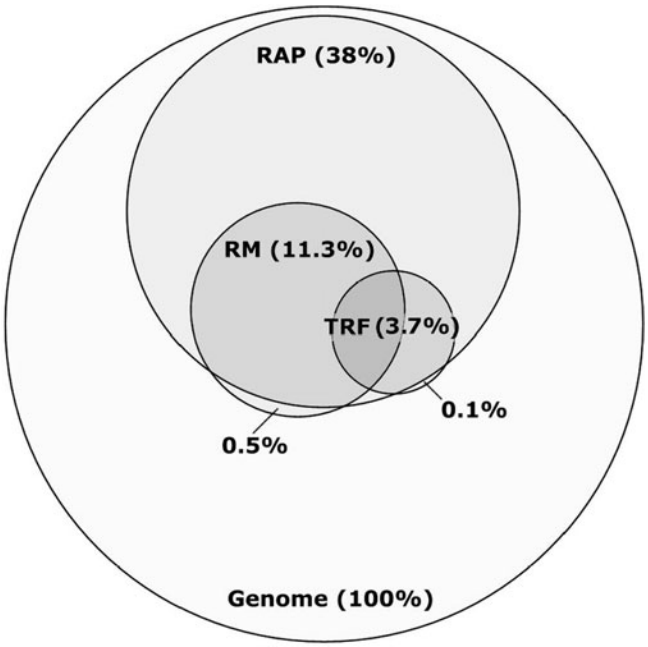

Fig. 6. Finding repeats in *C.elegans* genome. RAP identifies 38% the genome as repetitive, RepeatMasker (RM) 11.3% and TandemRepeatFinder (TRF) 3.7%. Only a small fraction of the sequences recognized by RM and TRF were not recognized by RAP (0.5 and 0.1%, respectively). The following parameters were used for the RAP program: 16 bases word length, 200 gapped/ungapped patterns and 300 counts threshold in a single base window. A filter was set to lower the noise due to the different ending of the repeats. All the percentages refer to the whole genome.

RepeatMasker is only able to identify repeats that have already been characterized. Therefore, a comparison of the results obtained from the two programs does not measure the performance of RAP against RepeatMasker, but rather RAP against known repeats. Another point is that RAP does not answer the question whether a sequence is a repeat or not, but how much it is repetitive. Therefore, the concept of repeat is not qualitative as in the case of RepeatMasker ('Is this local sequence one of the specific repeats that I know?'), but quantitative ('How frequent in the genome are the gapped and ungapped words present in this local sequence?'). As a consequence, the level of repetitivity depends very much on the parameters used by the program.

Despite the above considerations, we have compared the results obtained by RAP with those of TandemRepeatFinder (Benson, 1999) and RepeatMasker. It can be seen (Fig. 6) that the sensitivity of RAP is very high, being able to detect the vast majority of the known repeats, while the small percentage of sequences not detected by RAP is often due to short extraneous subregion embedded within known repeats. Interestingly, RAP finds many highly repetitive regions that are not detected either by RepeatMasker or by TandemRepeatFinder. From a preliminary analysis, these regions correspond in many cases to paralogous genes and other repetitive elements that have not yet been characterized.

Considering that the genome of *C.elegans* is ~100 Mb long and that there are 134 225 920 different dsDNA words of 14 bases, we can conclude that a word length producing a number of possible dsDNA words close to the length of the genome (redundancy ≈ 1) is suitable for the identification of repeated regions. However, words of 12 bases produce informative results (Fig. 5). Since there are only 8 390 656 possible dsDNA words of 12 bases, in this case the redundancy is only 0.08. Therefore, we think that any value of redundancy between 0.1 and 1 should be suitable for an informative analysis.

If we consider a typical mammalian genome of 3 Gb, a word length of 14 bases would probably be too short as it corresponds to a redundancy of ~0.05. However, a word length of 16 bases (2 147 516 416 different dsDNA words) will give a redundancy of 0.7, which should produce very good results. Therefore, we can predict that RAP will be suitable to perform repeat analysis in mammalian genomes using a word length of 16 bases.

In conclusion, the algorithm presented in this paper when, compared with other word counting algorithms, offers the possibility to increase the sensitivity and resolution of repeat analysis by performing multiple reading at the same genomic position, including gaps in the counted words. This strategy, in combination with the new data structure for counting dsDNA words, has proved very effective for the purpose of *de novo* identification of repeats.

ACKNOWLEDGEMENT

The authors wish to acknowledge the support of MIUR, Progetto Finalizzato 'Genomica Funzionale'. D.C. was supported by the Italian Telethon Foundation (Grant B57.1), N.C. is in part supported by the grant FIRB-RBNE01F5WT_007.

REFERENCES

Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Bao,Z. and Eddy,S.R. (2002) Automated *de novo* identification of repeat sequence families in sequenced genomes. *Genome Res.*, **12**, 1269–1276.

Bedell,J.A., Korf,I. and Gish,W. (2000) MaskerAid: a performance enhancement to RepeatMasker. *Bioinformatics*, **16**, 1040–1041.

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

Burrows,M. and Wheeler,D.J. (1994) A block sorting lossless data compression algorithm. *Technical Report 124*, Digital Equipment Corporation, Palo Alto, CA.

Healy,J., Thomas,E.E., Schwartz,J.T. and Wiegler,M. (2003) Annotating large genomes with exact word matches. *Genome Res.*, **13**, 2306–2315.

Jurka,J., Walichiewicz,J. and Milosavljevic,A. (1992) Prototypic sequences for human repetitive DNA. *J. Mol. Evol.*, **35**, 286–291.

Jurka,J., Klonowski,P., Dagman,V. and Pelton,P. (1996) CENSOR—a program for identification and elimination of repetitive elements from DNA sequences. *Comput. Chem.*, **20**, 119–122.

Kent,W.J., Sugnet,C.W., Furey,T.S., Roskin,K.M., Pringle,T.H., Zahler,A.M. and Haussler,D. (2002) The human genome browser at UCSC. *Genome Res.*, **12**, 996–1006.

Kurtz,S. (1999) Reducing the space requirement for suffix trees. *Software Pract. Esperince*, **29**, 1149–1171.

Kurtz,S., Choudhuri,J.V., Ohlebusch,E., Schleiermacher,C., Stoye,J. and Giegerich,R. (2001) REPuter: the manifold applications of repeat analysis on a genomic scale. *Nucleic Acids Res.*, **29**, 4633–4642.

Lefebvre,A., Lecroq,T., Dauchel,H. and Alexandre,J. (2002) FORRepeats: detects repeats on entire chromosomes and between genomes. *Bioinformatics*, **19**, 319–326.

Manber,U. and Myers,E.W. (1993) Suffix array: a new method for on-line string searches. *SIAM Journal of Computing*, **22**, 935–948.

McCreight,E.M. (1976) A space-economical suffix tree construction algorithm. *J. Algorithms*, **23**, 262–272.

Valle,G. (1993) Discover 1: a new program to search for unusually represented DNA motifs. *Nucleic Acids Res.*, **21**, 5152–5156.

Volfovsky,N., Haas,B.J. and Salzberg,S.L. (2001) A clustering method for repeat analysis in DNA sequences. *Genome Biol.*, **2**, RESEARCH0027.