BIOINFORMATICS APPLICATIONS NOTE Vol. 30 no. 18 2014, pages 2662–2664
doi:10.1093/bioinformatics/btu312

Sequence analysis

Advance Access publication May 27, 2014

keeSeek: searching distant non-existing words in genomes for PCR-based applications

Marco Falda[1], Paolo Fontana[2], Luisa Barzon[1], Stefano Toppo[1,*] and Enrico Lavezzo[1]

[1]Department of Molecular Medicine, University of Padova, Padova, I-35131, Italy and [2]Department of Computational Biology, Edmund Mach Foundation, S. Michele All'Adige, I-38010 (TN), Italy

Associate Editor: Alfonso Valencia

ABSTRACT

Summary: The search for short words that are absent in the genome of one or more organisms (neverwords, also known as nullomers) is attracting growing interest because of the impact they may have in recent molecular biology applications. keeSeek is able to find absent sequences with primer-like features, which can be used as unique labels for exogenously inserted DNA fragments to recover their exact position into the genome using PCR techniques. The main differences with respect to previously developed tools for neverwords generation are (i) calculation of the distance from the reference genome, in terms of number of mismatches, and selection of the most distant sequences that will have a low probability to anneal unspecifically; (ii) application of a series of filters to discard candidates not suitable to be used as PCR primers. KeeSeek has been implemented in C++ and CUDA (Compute Unified Device Architecture) to work in a General-Purpose Computing on Graphics Processing Units (GPGPU) environment.

Availability and implementation: Freely available under the Q Public License at http://www.medcomp.medicina.unipd.it/main_site/doku.php?id=keeseek

Contact: stefano.toppo@unipd.it

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on February 17, 2014; revised on April 16, 2014; accepted on April 28, 2014

1 INTRODUCTION

In the recent past, different methods for finding non-existing sequences in genomes, previously called 'nullomers' or 'unwords', have been proposed (Garcia *et al.*, 2011; Hampikian and Andersen, 2007; Herold *et al.*, 2008; Wu *et al.*, 2010). Though nullomer definition is accepted, the term may be confusing and evocative of a k-mer of 'null' length. For this reason, we propose 'neverwords' to indicate strings that do not exist in a genome. Such neverwords have been proposed for the following aims: (i) studies of population genetics, species identification and evolution; (ii) drug discovery and development; (iii) target design for recalling or eliminating genetically engineered organisms (e.g. suicide targets); (iv) design of molecular barcodes (Goswami *et al.*, 2013) or specific adaptors for PCR primers [e.g. for the detection of viral insertion sites in host's genomes (Gabriel *et al.*, 2009)]. The algorithms proposed so far for neverwords generation are only focused on the detection of absent words in genomes, without providing any information about their distance in terms of number of mismatches. This additional information is crucial when such k-mers are used as barcodes or PCR primers: for these purposes, ideal neverwords must be distant enough to any position of the reference genome and must possess 'primer-like' features.

Here we present keeSeek, software developed for the design of distant k-mers that can be used as barcodes or PCR primers. keeSeek allows the user to generate, for any reference genome, a set of k-mers absent in that genome, selecting a desired length and a minimum number of mismatches along all positions of the reference. The advantage versus previous tools is the capability to produce longer sequences (results were obtained up to the length of 31 nt) and, most importantly, to provide information about the number of mismatches and the position of the best match in the reference. In addition, a set of filters has been implemented to select only k-mers with primer-like features. The algorithm is written in C++ and CUDA, to exploit the high parallelization provided by Graphics Processing Units (GPUs).

2 METHODS

K-mers generation: Here we report the example and test of 20-mers. Because the number of different k-mers of length l that can be generated using four symbols is 4^l, the amount of 20-mers that must be tested when looking for the most distant candidate compared with a reference genome is in the order of 10^{12}. By default, keeSeek starts with the generation of all the possible k-mers of a defined length. Alternatively, the user can define a priori the nucleotide composition of the k-mer: if the goal is a primer sequence, a good balance among the different nucleotides should provide better chances to generate complex k-mers. Starting from the base composition selected by the user, keeSeek will generate and test all possible anagrams that can be obtained by permutating nucleotides. The total number of combinations of a 20-mer made of 5A, 5T, 5C and 5G drops from 4^{20} to 11 732 745 024.

K-mers filtering: for some purposes, as primer design, the reduction of the combinatorial complexity is obtained by filtering k-mers according to a set of criteria, instead of testing all of them. Filtering is performed on the basis of sequence complexity, conformity of 3′-terminals, detection of long homopolymers, melting temperature and the tendency to generate hairpins and duplexes (see details on Supplementary File S1.3). These filters are applied by default in the 16–26 range of lengths, but they can be disabled if the aim is the exhaustive generation of neverwords. After the filtering step, potential candidate 20-mers of composition 5A-5T-5C-5G are further reduced to 139 090 688.

K-mers evaluation: each k-mer is searched against both strands of the reference genome, to evaluate its presence/absence and the minimum

*To whom correspondence should be addressed.

Table 1. Examples of nullomers of length 20 generated for three different reference genomes

Reference genome	Nullomer	Minimum number of mismatches	Generation time (min:sec:msec)
H.sapiens (GRCh37.64 ENSEMBL release)	ACTTAGATTGACGCGCATCG	4	2:08:142
	CACTTCGAGCAAGTTAGCTG	3	2:08:101
	CAAGTGCGAACGTTTCGTCA	3	2:13:794
A.thaliana (NC_003070.9, NC_003071.7, NC_003074.8, NC_003075.7, NC_003076.8)	GAACCTCGAACTATGGTTGC	4	0:05:443
	CATACCTGAATCGTGAGTCG	4	0:05:416
	GAGTGCATTATCGCTACGAC	4	0:05:420
M.tuberculosis (NC_000962.2)	CTACGTATGCAAGGCTTCAG	5	0:00:237
	CTGTCTGAATGCCACATGGA	5	0:00:205
	CTCTGTTTAGACGACCAGGA	5	0:00:208

Note: Each nullomer was generated in an independent run of the algorithm, using the 'anagram mode': we started by defining an equal distribution of the four nucleotides (5A, 5T, 5C and 5G), and we reshuffled the anagrams starting from a random seed (-R 0). One of the nullomers with the highest distance from the respective reference genome is shown for each run. Times refer to the first block of 128 evaluated k-mers (see Supplementary File S1.6 for details) obtained on a Tesla GPU M2050. Loading of the genome is not considered.

number of mismatches. Because the tool is aimed at primer-like k-mers, the search is ungapped, allowing the implementation of an algorithm linear in space and time (see Supplementary File S1.1 for details). We encode both the reference and the k-mer using a compressed binary representation in blocks of 64 bits and, by using a sliding window, each candidate is then compared with all reference positions within the blocks; because the target architectures provide 64 bit registers, differences can be computed using fast bitwise operators. By combining the minimal number of mismatches among all the blocks through a multithreaded reduction, the overall minimal distance between the k-mer and the genome is calculated (see details on Supplementary File S1.1). This step is computationally intensive and has been optimized to run on GPUs, taking advantage of hundreds or thousands computing cores. Because the number of k-mers to test is huge (exponential in k), the anytime nature of the algorithm provides the user valuable suboptimal solutions in reasonable time. K-mer generation is memoryless but ordered, so that the tool can be stopped and restarted from the last generated k-mer, continuing the scan of new candidates.

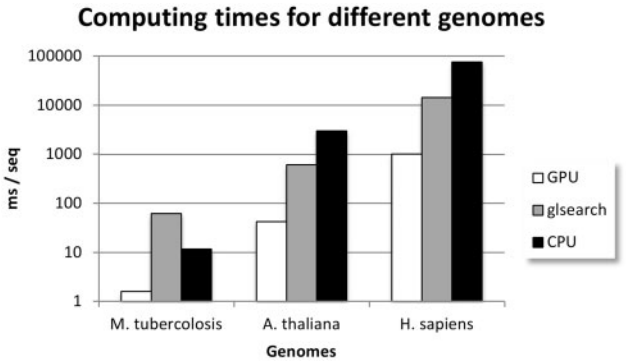

Fig. 1. Computational times required to search for a sequence of length 20 on three selected genomes, obtained by running the GPU implementation of keeSeek (default options), its multithreaded CPU version (-N option) and glsearch algorithm. The *y*-axis is in logarithmic scale. See Supplementary File S1.7 for additional details

3 RESULTS

We have assessed keeSeek on three model organisms, *Mycobacterium tuberculosis*, *Arabidopsis thaliana* and *Homo sapiens*, and at different word lengths, up to 31 nt. Table 1 reports, for each reference genome, a small panel of 20-mers with a minimum distance of three mismatches and the relative computational time required to generate them.

The results were validated by searching them with a third-party tool, glsearch (Pearson, 2000), based on the Needleman and Wunsch algorithm (Needleman and Wunsch, 1970). Additionally, a comparison of searching times between GPU and CPU implementations has been performed (Fig. 1).

4 CONCLUSIONS

keeSeek is a tool developed for the generation of non-existing words we call neverwords with a minimum selectable distance from a reference sequence. The tool is optimized for the production of PCR primer-like sequences of different lengths suitable for use in molecular biology. keeSeek can also find useful applications in experiments of targeted genomic manipulation such as those based on zinc finger nucleases, Transcription Activator-Like Effector Nucleases (TALEN) and Clustered Regularly Interspaced Short Palindromic Repeats (CRISPRs) (Gaj *et al*., 2013).

ACKNOWLEDGEMENTS

The authors thank the LICC Laboratory headed by Prof. A. Polimeno and CRIBI Genomics headed by Professor G. Valle for allowing us to develop on Tesla GPUs. The authors thank Prof. G. Valle for critical reading of the article.

Funding: This research is supported by PRAT CPDA138081/13 from University of Padova.

Conflicts of Interest: none declared.

REFERENCES

Gabriel,R. *et al*. (2009) Comprehensive genomic access to vector integration in clinical gene therapy. *Nat*. *Med*., **15**, 1431–1436.

Gaj,T. *et al*. (2013) ZFN, TALEN, and CRISPR/Cas-based methods for genome engineering. *Trends Biotechnol*., **31**, 397–405.

Garcia,S.P. *et al*. (2011) Minimal absent words in prokaryotic and eukaryotic genomes. *PLoS One*, **6**, e16065.

Goswami,J. *et al*. (2013) Safeguarding forensic DNA reference samples with nullomer barcodes. *J*. *Forensic Leg*. *Med*., **20**, 513–519.

Hampikian,G. and Andersen,T. (2007) Absent sequences: nullomers and primes. *Pac*. *Symp*. *Biocomput*., **12**, 355–366.

Herold,J. *et al*. (2008) Efficient computation of absent words in genomic sequences. *BMC Bioinformatics*, **9**, 167.

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J*. *Mol*. *Biol*., **48**, 443–453.

Pearson,W.R. (2000) Flexible sequence similarity searching with the FASTA3 program package. *Methods Mol*. *Biol*., **132**, 185–219.

Wu,Z.D. *et al*. (2010) Efficient computation of shortest absent words in a genomic sequence. *Inf*. *Process*. *Lett*, **110**, 596–601.