|  |
| --- |
| Genetics and population analysis  IEPSM: An improved exact string matching method for genomic sequencing data  Lin Dai1,\*, Li Wang1 and Zhang Zhang 2,\*  1 Department of Computer Science, Beijing Institute of Technology, Beijing, China.  2 CAS Key Laboratory of Genome Sciences and Information, Beijing Institute of Genomics, Chinese Academy of Sciences, Beijing, China  Received on XXXXX; revised on XXXXX; accepted on XXXXX  Associate Editor: XXXXXXX |

[[1]](#footnote-2)\*abstract

**Summary:** String matching algorithm plays a vital role in bioinformatics. Here we present IEPSM, an improved exact packed string matching algorithm that is dedicated for biological sequences. IEPSM features optimized word-size packed strings and adopts a big hash value to decrease byte-by-byte comparisons. Comparative results on multiple empirical datasets show that IEPSM achieves better efficiency by comparison with existing algorithms. Thus, IEPSM is of broad utility for searching a specific pattern in the era of big biological data.

**Availability and implementation:** IEPSM is available through Bioconductor. It is released under the GPL-2 license. [link]

**Contact:** dailiu@bit.edu.cn

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

# introduction

String matching is an important problem that has been thoroughly studied in computer science, with broad applications in bioinformatics as well as natural language processing, information retrieval, etc. For example, it is used to find similar sequence or locate a segment in a long sequence (Senapati, Sandip and Sahoo, 2012). Currently, several string matching algorithms are used on biological sequences, such as tvsbs (Thathoo, Virmani, Lakshmi, Balakrishnan and Sekar, 2006), graspm (Deusdado and Carvalho, 2009) etc. With the rapid development of high-throughput sequencing technologies, it has become easier and cheaper to obtain vast quantities of biological sequences, accordingly posing great challenges in searching for a specific pattern (viz., sequence or segment) within a large volume of biological sequences. Therefore, it is of fundamental importance to design more effective string matching algorithms to address this challenge (Eric, Leena and Jorma, 2011).

There are several algorithms that have developed for exact string matching in the past years (Navarro and Raffinot, 2002). Among them, an algorithm called EPSM (Faro and Kulekei, 2013) obtained better performance by comparison with others. It uses exact packed string matching technique (Frediksson, 2002), in which multiple characters are packed into one block-character, so that the characters can be compared in bulk rather than individually. EPSM computes fingerprint values by a hash function using SingleInstruction Multiple Data instructions, which supports parallel execution of some operations via a set of special instructions. However, EPSM is developed for general purposes, without considering features of biological sequences; the max shift distance of EPSM algorithm is *m*-8, where *m* is the length of the given pattern, and it is not an optimal shift distance for biological sequences. [add one more sentence to describe why it is not optimal?] Here we take good account of biological sequences features and propose an improved EPSM (IEPSM) by adopting more shift distances and less byte-by-byte comparison calls.

# methods

EPSM algorithm stores all the 8 bytes substrings of the pattern and their position in a hash table. In the searching phase, it checks the last 8 bytes of the current searching window using the hash table to find possible matches. It then checks all the possible positions byte-by-byte. Since the SIMD instructions are used to calculate hash value of packed string, the size of packed strings is set as 8 bytes. When finishing the examination of current window, it jumps forward by *m*-8 bytes to perform next examination, where *m* is the length of the pattern.

IEPSM is developed based on EPSM with specialized improvements for string matching in biology sequences. Specially, its basic idea to achieve higher efficiency for matching a specific pattern in biological sequences is to get larger jumping distance and less byte-by-byte comparisons. Our improvements are as follows.

1) Optimizing the size of packed strings for biological sequences. In fact, shift distance of EPSM is *m*-*B*, where *B* is the size of packed strings. If a smaller *B* is adopted, we can achieve larger shift distance. In the meantime, however, smaller *B* also results in more hash conflicts and more possible matches, which incur more additional examinations. We decide the optimal *B* experientially.

2) Reducing byte-by-byte comparison between pattern and text. EPSM uses a 32bit hash value and then mod the size of hash table to determine the entry of hash table and the entrance of byte-by-byte comparison. Intuitively, taking bigger numbers as the entrance condition will trigger less byte-by-byte comparison. We use 64bit numbers as the entrance condition. In practice, we add a new field *fingerprint* to the definition of the structure of hash table node. These fingerprints are initialized during pre-processing phase. Like EPSM, we use SIMD instructions to quickly calculate hash value of substrings, thus 8 bytes substrings are used to generate the fingerprint. In searching phrase, before calling *memcmp* to perform byte-by-byte comparison, the algorithm examines the 64bit fingerprint of current window against that of the pattern. By this way, we avoid lots of *memcmp* calls.

# results

## Experimental Settings

IEPSM was implemented using C. All experiments were conducted on a PC with Intel(R) Xeon E3-1230 V2 at3.30GHz, 4G memory, running Linux Mint 13. The size of the hash table in IEPSM equals to that of EPSM algorithm, i.e., 2048.

We compared IEPSM against five state-of-art string matching algorithms, viz., Tvsbs (Thathoo, Virmani, Lakshmi, Balakrishnan and Sekar, 2006), Ufndmq (Durian, Holub, Peltola and Tarhio, 2009), Hashq (Lecroq, 2007), Fsbndmq (Peltola and Tarhio, 2011) and EPSM (Faro and Kulekei, 2013). All the algorithms were tested on four different data sets, i.e., *Escherichia coli*, *Rice*, *human genome*, *coli amino acid* (ftp). ~~To make the experimental results more comparable, we make the data sets with the same size through simple duplication.~~ The pattern string is randomly extracted from the text. Because the max para length of SSE function is 8 bytes, we choose B between 4 and 8.

## Optimal packed string size

The optimal packed string size is essential to the algorithm’s performance. The optimal size would be affected by the length of pattern and the alphabet of the sequences. To determine the optimal size, we tested different lengths of packed string on different pattern sizes on different data sets.

The experimental results show that the optimal packed string size for gnome sequences is 6 when *m* is less than 40, otherwise it is 8. And the optimal packed string size for protein sequences is 4 when *m* is less than 256, and otherwise it is 8. In the following experiments, we use these optimal parameters

Based on my understanding, the point is:

1. Nucleotide sequences: if m<16, then B=4; if m<40, then B=6; if m>40, then B=8

2. Protein sequences: if m<256, then B=4; if m>256, then B=8.

These optimized B values are very important to IEPSM and should provide more details.

1. Optimized B based only on database S1/S4 is not enough, which should be obtained on three datasets in nucleotide and proteins sequences, respectively: E. coli, Rice and Human. Make all these results as supplementary files.

(1) Nucletodie sequences:

E.coli:

<ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria/Escherichia_coli_K_12_substr__MG1655_uid57779/NC_000913.fna>

human:

<ftp://ftp.ensembl.org/pub/current_fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz>

Rice:

<ftp://ftp.ensemblgenomes.org/pub/plants/release-24/fasta/oryza_sativa/dna/Oryza_sativa.IRGSP-1.0.24.dna.genome.fa.gz>

(2) protein sequences:

E.coli:

<ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria/Escherichia_coli_K_12_substr__MG1655_uid57779/NC_000913.faa>

human:

<ftp://ftp.ensembl.org/pub/current_fasta/homo_sapiens/pep/Homo_sapiens.GRCh38.pep.all.fa.gz>

Rice:

<ftp://ftp.ensemblgenomes.org/pub/plants/release-24/fasta/oryza_sativa/pep/Oryza_sativa.IRGSP-1.0.24.pep.all.fa.gz>

2. How different patterns generated?3. Does B>8 make sense?

## Efficiency

We evaluated IEPSM by comparing it to five different algorithms on four data sets (Fig. 1*)*. It is evident that across all tested data sets, IEPSM outperforms other algorithms by achieving less matching time.

E:\Documents\human_dna.tifE:\Documents\human_dna.tif

Fig.1. Comparisons of IEPSM against five popular algorithms on four data sets. The X axis is the length of the pattern and the y axis represents the matching time [similar as Section 3.2, generate 6 figures and we may use human nucleotide and human protein as example in Fig. 1 and the rest four are put into supplementary files. Do not need to make these datasets with same sizes].

The advantage of the new algorithm decreases when the length of patterns grows. When *m* > 40 on genome datasets and *m* > 256 on protein datasets, IEPSM and EPSM take the same size of packed string. Thus the performance improvement is attributable to the filtering by using big fingerprints. This filter avoids lots of this kind of comparison. When pattern strings are short, the performance improvement also benefits from the increment of shift distances. For example, on gnome dataset, when *m* is 12, the shift distance of EPSM is 4, but that of IEPSM could be 6. But when *m* is larger than 40, the shift distance of the new algorithm is the same as EPSM. The shorter the pattern string is, IEPSM is capable of getting larger shift distance than EPSM. Therefore, in the experiments with short pattern string, IEPSM outperforms other algorithms by obtaining extraordinary improved performance.

At the same time, we can observe that IEPSM and EPSM obviously outperform other algorithms. This is because the use of SIMD instructions and these results are consistent with the literature (Faro and Kulekei, 2013).

# Conclusions

We presented an improved algorithm based on EPSM for string matching problem on biological sequences. Based on characteristics of biological sequences, IEPSM took the optimized size of the packed string to generate the hash table. Besides, IEPSM used a 64bit number as a fingerprint of string, the fingerprint was generated by 8 bytes string. We used the fingerprint filtering many comparisons before taking byte-by-byte comparison. And the experiment results proved that by using these two optimized strategies, IEPSM is more efficient than EPSM.

acknowledgements

*Funding*: National Programs for High Technology Research andDevelopment (863 Program; 2014AA021503 and 2012AA020409) and the “100-Talent Program” of Chinese Academy of Sciences.

References

K.K.Senapati, Sandip Mal, G.Sahoo. (2012)RS-A Fast Pattern Matching Algorithm for Biological Sequences. International Journal of Engineering and InnovativeTechnology (IJEIT) Volume 1, Issue 3.

R. Thathoo and A. Virmani and S. S. Lakshmi and N. Balakrishnan and K. Sekar. (2006)TVSBS: A Fast Exact Pattern Matching Algorithm for Biological Sequences. J. Indian Acad. Sci., Current Sci., vol.91, n.1, pp.47—53.

S. Deusdado and P. Carvalho. GRASPm. (2009)An efficient algorithm for exact pattern-matching in genomic sequences. Int. J. Bioinformatics Res. Appl., vol.5, n.4, pp.385—401.

Eric Rivals, Leena Salmela, and Jorma Tarhio. (2011)EXACT SEARCH ALGORITHMS FOR BIOLOGICAL SEQUENCES. Algorithms in Computational Molecular Biology: Techniques, Approaches and Applications, John Wiley & Sons, Inc. (Ed.) 91-111.

G. Navarro, M. Raffinot, Flexible Pattern Matching in Strings Practical On-Line Search Algorithms for Texts and Biological Sequences, Cambridge University Press, 2002.

S. Faro and M. O. Kulekci. (2013)Fast Packed String Matching for Short Patterns. Meeting on Algorithm Engineering and Experiments.

K. Fredriksson. (2002)Faster string matching with super-alphabets. String Processing and Information Retrieval, Springer, pp. 207–214.

B. Durian and J. Holub and H. Peltola and J. Tarhio. (2009)Tuning BNDM with q-Grams. Proceedings of the Workshop on Algorithm Engineering and Experiments, pp.29--37.

T. Lecroq. (2007)Fast exact string matching algorithms. Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, ipl, vol.102, n.6, pp.229--235.

H. Peltola and J. Tarhio. (2011)Variations of Forward-SBNDM. Proceedings of the Prague Stringology Conference. Czech Technical University, Prague, Czech Republic, 11, pp.3--14.

ftp://ftp.ncbi.nlm.nih.gov/genbank/genomes/Bacteria/.

1. \*To whom correspondence should be addressed. [↑](#footnote-ref-2)