Biotechnology & Biotechnological Equipment, 2014
Vol. 28, No. 5, 958–967, http://dx.doi.org/10.1080/13102818.2014.959711

Taylor & Francis
Taylor & Francis Group

ARTICLE; BIOINFORMATICS

Towards computational improvement of DNA database indexing and short DNA query searching

Done Stojanov*, Sašo Koceski, Aleksandra Mileva, Nataša Koceska and Cveta Martinovska Bande

Faculty of Computer Science, Department of Computer Technologies and Intelligent Systems, University "Goce Delčev", Štip, Republic of Macedonia

(*Received 31 October 2013; accepted 9 July 2014*)

In order to facilitate and speed up the search of massive DNA databases, the database is indexed at the beginning, employing a mapping function. By searching through the indexed data structure, exact query hits can be identified. If the database is searched against an annotated DNA query, such as a known promoter consensus sequence, then the starting locations and the number of potential genes can be determined. This is particularly relevant if unannotated DNA sequences have to be functionally annotated. However, indexing a massive DNA database and searching an indexed data structure with millions of entries is a time-demanding process. In this paper, we propose a fast DNA database indexing and searching approach, identifying all query hits in the database, without having to examine all entries in the indexed data structure, limiting the maximum length of a query that can be searched against the database. By applying the proposed indexing equation, the whole human genome could be indexed in 10 hours on a personal computer, under the assumption that there is enough RAM to store the indexed data structure. Analysing the methodology proposed by Reneker, we observed that hits at starting positions $p \leq k - |q|$ are not reported, if the database is searched against a query shorter than k nucleotides, such that k is the length of the DNA database words being mapped and $|q|$ is the length of the query. A solution of this drawback is also presented.

Keywords: DNA database; fast indexing and search; all hits; *E. coli*

Introduction

Increased knowledge about the complex human genome is revealing its importance and impact on people's lives. The advances in computer science have contributed to the storage, dissemination, search and analysis of human genome data with increased efficiency and accuracy. That is why genetic databases are gaining more and more popularity in the research community today. Genetic databases often include entire genomes and could be used for searching particular sequences in genetic disease analysis, DNA fingerprinting, genetic genealogy or analysis of short sequences, such as non-standard codon structure [1] and codon context frequency.[2]

As soon as the first genetic database became available on the Internet, the necessity of fast DNA database-processing algorithms became a challenge that is still a challenge to researchers today. Being database inapplicable, dynamic programming-based solutions for global/local sequence alignment, such as Needleman–Wunsch [3] and Smith–Waterman,[4] have been substituted with faster, heuristic seed-based algorithms such as FASTA (Fast Alignment) [5] and BLAST (Basic Local Alignment Search Tool),[6] performed in two phases. In the first, so-called preprocessing phase, the matching positions of highly similar regions are identified as seeds, and in the second phase, the seeds are extended to local alignment. Usually, not every initial seed is extended to full alignment; instead, many of them are discarded by filtering, which results in lower run-time.

One common task – searching a genetic database to find exact matches for a non-degenerate or partially degenerate query – is usually done by using web applications hosted and run on remote web servers.[7] For smaller databases, computer desktop programs can be also used, with all the data kept in the main memory. The main feature of all these algorithms and tools is the phase of database indexing, which precedes and speeds up the actual searching phase. There are also algorithms for searching large genetic databases rapidly on desktop computers with limited RAM, like MICA (K-Mer Indexing with Compact Arrays),[8] which stores indexed data on a disk and retrieves relevant data selectively during the searching phase. Indexing a DNA sequence with MICA is achieved by dividing the sequence in chunks of $(2^{16}-1)$ bases, scanning each chunk with a window of width K and storing the positions of all overlapping K-mers in array.

One group of algorithms uses suffix trees (OASIS [9] and the three versions of MUMmer [10–12]), and

*Corresponding author. Email: done.stojanov@ugd.edu.mk

enhanced suffix arrays (ESAs) (Vmatch [13]) for building database indexes. ESAs consist of four arrays (suffix, longest common prefix, child and suffix link arrays) that together reach the full expressiveness of suffix trees. Suffix trees and ESAs are time efficient with complexity of O(n) and space complexity of O(n log(n)). The Kurtz [14] implementation of suffix tree requires 17.25 bytes per base, which is translated to 50 GB for the human genome, while ESAs require 4−8 bytes per base. There are also sparse suffix arrays (SSA), which index every *K*-th (sparseness factor) suffix of the sequence.[15] Their sparseMEM tool is able to find maximal exact matches faster than the previous methods, while using less memory. The tool essaMEM [16] optimizes the previous method by supplementing SSAs with a sparse child array for large sparseness factors, and this is known as enhanced sparse suffix array (ESSA).

Other new strategies deploy compressed indexing techniques that reduce the space complexity to O(n) bits. Techniques include Ferragina−Manzini index (FM-index),[17] compressed suffix arrays (CSA) index [18] and Burrows−Wheeler transform (BWT) index.[19] For DNA sequences, BWT indexing was found to be the most efficient, and the memory requirement is less than 0.3 bytes per base. For the human genome, this requires only 1 GB memory and the whole index can reside in the main memory of a personal computer (PC). CSA index implemented with the BWT is used in [20,21] while BWT index is used in BWT-SW.[22] Another tool, backwardMEM, [23] uses enhanced CSA by indexing each *K*th suffix array value.

Benson [24] has proposed an algorithm for identification of all tandem repeats, without having to specify the pattern. Neglecting patterns that occur within a few database sequences, TEIRESIAS [25] is able to generate all maximal patterns that appear within at least a (a is user-defined) sequences. RAPID [26] is a probabilistic word-searching approach. A different significance is assigned to each match of length k, depending of the number of occurrences of the match. By partitioning the query and the subject sequence in fragments of fixed size, referred as windows, sequence search tree (SST) [27] can identify approximate matches, in time proportional to the logarithm of the database size.

Hash-based indexing strategies (SSAHA (Sequence Search and Alignment by Hashing Algorithm) [28,29] and BLAT (BLAST-like alignment tool) [30]), which are currently in more widespread use for DNA databases, require 1 byte or less per base, and they can be orders of magnitude faster than FASTA or BLAST, which index the query sequence rather than the database. SSAHA [28] partitions subject database sequence into non-overlapping words of length k (k-mers), being mapped into numbers according to the SSAHA conversion function. BLAT [30] builds up an index of non-overlapping k-mers and their positions in the database, excluding k-mers that occur too often from the index as well as k-mers containing ambiguity codes. During the search stage, three different strategies are used in BLAT, in order to find homologous regions: searching for perfect hits, allowing at least one mismatch between two hits and searching for multiple perfect matches which are in close proximity to each other. One problem with SSAHA and BLAT is their limitation, which rises from sacrificing the completeness for speed. They cannot detect matches with less than k bases.

Some conceptual and computational drawbacks of SSAHA have been solved by Reneker and Shyu.[29] According to Reneker and Shyu,[29] overlapping matches can be identified, if instead of indexing non-overlapping words of k bases, overlapping words of the same size are tracked in the indexed data structure. Reneker and Shyu [29] also pointed out that matches which are shorter than k bases can be identified as suffixes within some of the indexed words of k bases.

However, the previous concept is incomplete if some of the matches are located at the beginnings of the genetic sequences. In order to detect all matches, even the ones which are not reported in [29], we propose an improved searching methodology by integrating suffix search and prefix search that provide more exact matching.

From a computational viewpoint, we propose a computational upgrade of the indexing formula used by SSAHA and Reneker and Shyu,[29] which results in a k-fold speed-up of the indexing phase. The storage aspects were also improved due to the exclusion of redundant records from the indexed data structure. Instead of a hash table, a sorted dictionary indexed data structure is employed, which allows identification of all matches without having to scan all records, and hence, the better search time performance.

Materials and methods

Database indexing

Hash-based solutions, such as SSAHA and Reneker's improvements of SSAHA, are performed in two phases. In the first phase, DNA database words of k consecutive nucleotides $w : b_1 b_2 \ldots b_{k-1} b_k,\ b_j \in \Sigma = \{A, C, T, G\}$, $1 \le j \le k$, are mapped into numbers, applying a concrete base-mapping function. SSAHA and Reneker employ different base-mapping functions due to the fact that none of the nucleotides can be zero-mapped. According to SSAHA, adenine is zero-mapped, $f(A) = 0$. Since database words $w : b_1 b_2 \ldots b_{k-1} b_k$ are mapped in integers as $f(w : b_1 b_2 \ldots b_{k-1} b_k) = \Sigma_{j=1}^{k} f(b_j) \times 4^{j-1}$, words ending with different number of **A**'s could not be distinguished if SSAHA's base-mapping function is used. For instance, since the mapped value of the words CAA and CAAAA is equal, $f(CAA) = f(CAAAA) = f(C) \times 4^0 = f(C)$, and

they are assumed to be equal, which is incorrect. In order to overcome SSAHA's mapping inconsistency, Reneker and Shyu [29] proposed a modified base-mapping function:$f(A)=1,\ f(T)=2,\ f(G)=3,\ f(C)=4$ for distinction of words ending with different number of **A**'s.

The modified base-mapping function guarantees that a different number $f(w)$ will be assigned for a different word, but neither SSAHA nor Reneker and Shyu's algorithm has been improved in terms of the time complexity of the indexing phase. Indexing complete genomes or recomputing the indexed data structure, if the DNA data has been modified, is a time-demanding process that could last even a few days, if it is executed on a PC. In these cases, there is a computational necessity to reduce the time span of the indexing phase. Therefore, we propose a computational upgrade of the indexing formula used in SSAHA and Reneker and Shyu's algorithm that speeds up the indexing k-fold, such that k is the length of the DNA words being mapped, in comparison to SSAHA and Reneker and Shyu's algorithm.

The proposed computational upgrade is based on the following. Once the first word from the ith DNA database sequence $w_{i,1}: b_{i,1}b_{i,2}\ldots b_{i,k-1}b_{i,k}$ has been mapped, the mapped value of each successive word $w_{i,j+1}: b_{i,j+1}b_{i,j+2}\ldots b_{i,j+k}$ can be calculated from the previous one $w_{i,j}: b_{i,j}b_{i,j+1}\ldots b_{i,j+k-1}$, according to Equation (1). The equation is based on the fact that two successive words $w_{i,j}$ and $w_{i,j+1}$ have exactly $k-1$ common nucleotides. Thus, each word $w_{i,j+1}$ can be derived from the previous one $w_{i,j}$, by excluding the leftmost nucleotide $b_{i,j}(f(w_{i,j})-f(b_{i,j})$ in Equation (1)), and by shifting the common nucleotides for one position to the left (division by 4) and addition of a new base $b_{i,j+k}(+f(b_{i,j+k})\times 4^{k-1}$ in Equation (1)):

$$f(w_{i,j+1})=(f(w_{i,j})-f(b_{i,j}))/4+f(b_{i,j+k})\times 4^{k-1} \quad (1)$$

For instance, if AACTT... is a DNA sequence and $k=4$, once the first word of four nucleotides AACT has been mapped, $f(w_{i,1}:AACT)=f(A)\times 4^0+f(A)\times 4^1+f(C)\times 4^2+f(T)\times 4^3=197$, the mapped value of the following word $ACTT$ can be calculated from the previous by applying Equation (1), $f(w_{i,2}:ACTT)=(f(w_{i,1})-f(A))/4+f(T)\times 4^3=49+2\times 64=177$ and so on.

Applying the mapping formula used in SSAHA and Reneker and Shyu's algorithm, $f(w: b_1b_2\ldots b_{k-1}b_k)=\sum_{j=1}^{k} f(b_j)\times 4^{j-1}$, $4k$ operations are performed in order to map a single word. To map a DNA sequence S_i of n nucleotides, $n-k+1$ words have to be mapped, i.e. $4k(n-k+1)$ operations are performed. Since the length of the DNA sequence n is greater than k ($n \gg k$), approximately $4nk$ operations are performed in order to map all overlapping words of k nucleotides in S_i.

Applying the proposed solution based on Equation (1), $4k$ operations are performed to map the first word only. Since the execution of Equation (1) requires four operations: subtraction $(f(w_{i,j})-f(b_{i,j}))$, division $((f(w_{i,j})-f(b_{i,j}))/4)$, multiplication $(f(b_{i,j+k})\times 4^{k-1})$ and an addition $((f(w_{i,j})-f(b_{i,j}))/4+f(b_{i,j+k})\times 4^{k-1})$ per mapped successive word, and there are $n-k$ successive and overlapping words of k nucleotides in S_i, a total of $4k+4(n-k)=4n$ operations would have to be performed, which results in a k-fold increase in the speed of the indexing phase in comparison to SSAHA and Reneker and Shyu's approach.

SSAHA and the technique of Reneker and Shyu store indexed data differently. SSAHA precomputes a hash table of 4^k keys, such that k is the length of the DNA words being mapped. By generating 4^k keys, SSAHA guarantees a different key for each different DNA word. Since the hash table is stored in the main memory, which has limited capacity, the application of SSAHA when running on a PC is suitable for indexing small databases. However, due to the fast processor−RAM communication, the time aspects of the indexing and search phase are relatively satisfactory.

Opposite to SSAHA, the indexed data can be stored in a file, which is kept on the disk or server, with much more storage capacity than the main memory. This idea, which is suitable for indexing long DNA sequences such as the human genome, has been employed by Reneker and Shyu. [29] However, the additional transfer of data between the RAM and the disk, during the indexing and searching phase, will slow down the time performance.

The idea of precomputing a hash table with 4^k keys before tracking tuples (i,p), such that i is the index of the DNA sequence where the word comes from and p is its starting position, can be improved in terms of memory complexity. If a greater value is taken for k, the number of precomputed keys in the hash table is increased. Since some of the precomputed keys may point zero (i,p) tuples, part of the main memory may be unnecessarily reserved.

In order to avoid situations in which memory is wasted for keys pointing zero (i,p) tuples, we propose the indexed data structure to be constructed dynamically. By reading and mapping overlapping words of k nucleotides found in the database, one can be sure that each key in the indexed data structure will point at least one tuple (i,p).

For instance, if $k=8$ and the DNA word **TTTTCATT** is not contained in the database, then the key $f(TTTTCATT)$ would be unnecessarily generated and kept in the memory. On the contrary, if the indexed data structure is constructed dynamically, by reading and mapping only words which are found in the database, a corresponding record key $f(TTTTCATT)$ would not exist. This would contribute towards an optimization of the memory requirements.

If a short DNA database is indexed, by taking a large value for k, a significant part of the main memory would be wasted for keeping redundant keys corresponding to

words which were not found in the database. This conceptual drawback of SSAHA not only implies unnecessary storage cost, but also slows down the search of a DNA pattern, since redundant keys are also examined in the searching phase. By reducing the size of the indexed data structure, better search time performance is expected, which may be of particular importance if an indexed data structure that tracks genomic data has to be searched.

DNA pattern search

A database of DNA sequences is primarily searched as part of molecular biology studies, with the main purpose of finding genes, identification of regulatory sequences, predicting intron−exon structures of genes, analysis of short tandem repeats, pseudogenes recognition, finding restriction sites, etc. In some of the cases, such as prediction of a promoter sequence based on a known consensus element, biological information can be attached to the sequences and the database can be searched against relatively short DNA patterns.

In order to speed up the search of the indexed DNA database stored in the main memory, instead of a hash table, we use a sorted dictionary *SD* data structure. Due to the fact that records are sorted in ascending order of the keys, all hits can be identified without having to scan the entire indexed data structure, which is going to improve the search time complexity.

Given the set of short DNA patterns $Q = \{q_1, q_2, \ldots, q_{n-1}, q_n\}$ searched against the database, database words of $k = |q_{\max}| + 1$ nucleotides, such that $|q_{\max}|$ is the length of the longest query in Q, are mapped in keys pointing tuples (i,p), such that i is the index of the database sequence from where the word has been read and p is the starting position of the word in the sequence. In such constellations, all query hits can be found either as suffixes or as prefixes within key-mapped words of k nucleotides. Also, note that the sorted dictionary is constructed by applying the proposed computational upgrade of the indexing formula used in SSAHA and the method of Reneker and Shyu.

Pattern q hit at starting position $p > k - |q|$ in the ith DNA sequence, such that $|q|$ is the length of the pattern, is found as a suffix of a key-mapped word from the same sequence. For illustration, the query q : **AA** is found as a suffix in **TTAA**, being the second key-mapped word from the DNA sequence S_2:C**TTAA**C…, if $k = 4$. The exact starting position of the hit can be derived from the tuple $(i,p) = (2,2)$, pointed by the sorted dictionary key $f(TTAA) = 90$, by increasing p for $k - |q| = 4 - 2 = 2$, i.e. the tuple $(i, p + k - |q|) = (2, 2 + 2) = (2,4)$ is reported.

An extension of the DNA pattern q up to length k with a minimum value of conversion $q^{\text{suffix},min} = \underbrace{A \ldots A}_{k-|q|}\, q_1 \ldots q_{|q|}$ is obtained by adding $k - |q|$ **A**'s to the left side. By adding $k - |q|$ **C**'s also to the left side of the pattern, an extension of the same length, but with maximum value of conversion $q^{\text{suffix},max} = \underbrace{C \ldots C}_{k-|q|}\, q_1 \ldots q_{|q|}$ is obtained.

Each sorted dictionary ***key***, such that Equation (2) is satisfied, points tuple (tuples)(i,p) tracking words that contain the searched DNA pattern as a suffix. Hits are reported as $(i, p + k - |q|)$ tuples, such that (i,p) is a tuple pointed by a ***key*** that satisfies the following equation:

$$f(q^{\text{suffix},min}) \le key \le f(q^{\text{suffix},max}) \tag{2}$$

Each sorted dictionary ***key***, such that $key > f(q^{\text{suffix,max}})$, does not map a word that ends up with the searched DNA pattern. Based on this, all hits at starting positions $p > k - |q|$ can be found, until the first ***key***, such that $key > f(q^{\text{suffix,max}})$ is read. None of the sorted dictionary records, such that $key > f(q^{\text{suffix,max}})$, have to be considered. Thus, better search time performance is expected compared to SSAHA.

The main computational drawback of the improvements of SSAHA proposed by Reneker and Shyu [29] is the inability to find DNA patterns that are located at the very beginnings of the DNA sequences which were indexed. Namely, DNA pattern hits at starting positions $p \le k - |q|$, such that k is the length of the words being mapped and $|q|$ is the length of the searched DNA pattern, are not reported by the algorithm of Reneker and Shyu. This drawback might result in an incomplete identification of repetitive nucleotide sequences within telomeres if chromosomes are searched in a reverse direction, and in an inability to detect a key DNA pattern, such as promoter consensus, within short DNA reads.

Therefore, we propose a solution of the previous drawback, based on checking whether the sorted dictionary ***key*** satisfies Equations (3) and (4). If the value of the ***key*** is in the range between $f(q^{\text{prefix},min})$ and $f(q^{\text{prefix},max})$, such that $q^{\text{prefix},min}$ and $q^{\text{prefix},max}$ are obtained by addition of $k - |q|$ **A**'s and **C**'s to the right side of the searched DNA pattern $(q^{\text{prefix},min} = q_1 \ldots q_{|q|} \underbrace{A \ldots A}_{k-|q|}$, $q^{\text{prefix},max} = q_1 \ldots q_{|q|} \underbrace{C \ldots C}_{k-|q|}$) and the remainder when dividing $key - f(q)$ by $4^{|q|}$ equals zero, then tuple (tuples) (i,p) pointed by the ***key***, track mapped words that contain the searched DNA pattern as a prefix. Reported tuples (i,p) identify DNA pattern hits unreported by the algorithm of Reneker and Shyu:

$$f(q^{\text{prefix,min}}) \le key \le f(q^{\text{prefix,max}}) \tag{3}$$

$$\text{mod}(key - f(q), 4^{|q|}) = 0 \tag{4}$$

The flowcharts of the indexing and searching phase are shown in Figures 1 and 2.

System characteristics

The proposed computational and conceptual upgrades were implemented in C# and tested on a Fujitsu Siemens computer with Core(TM) 2 Duo CPU at 2.67 GHz and 2 GB RAM.

Database used

To evaluate the *computational*, *storage* and *matching* performances of the proposed approach, different *Escherichia coli* DNA fragments of a total size of 0.1 Gb (giga bases) retrieved from the European Nucleotide Archive, [31] were indexed and the result was compared to that obtained by SSAHA and Reneker and Shyu's algorithm, using the same computational resources.

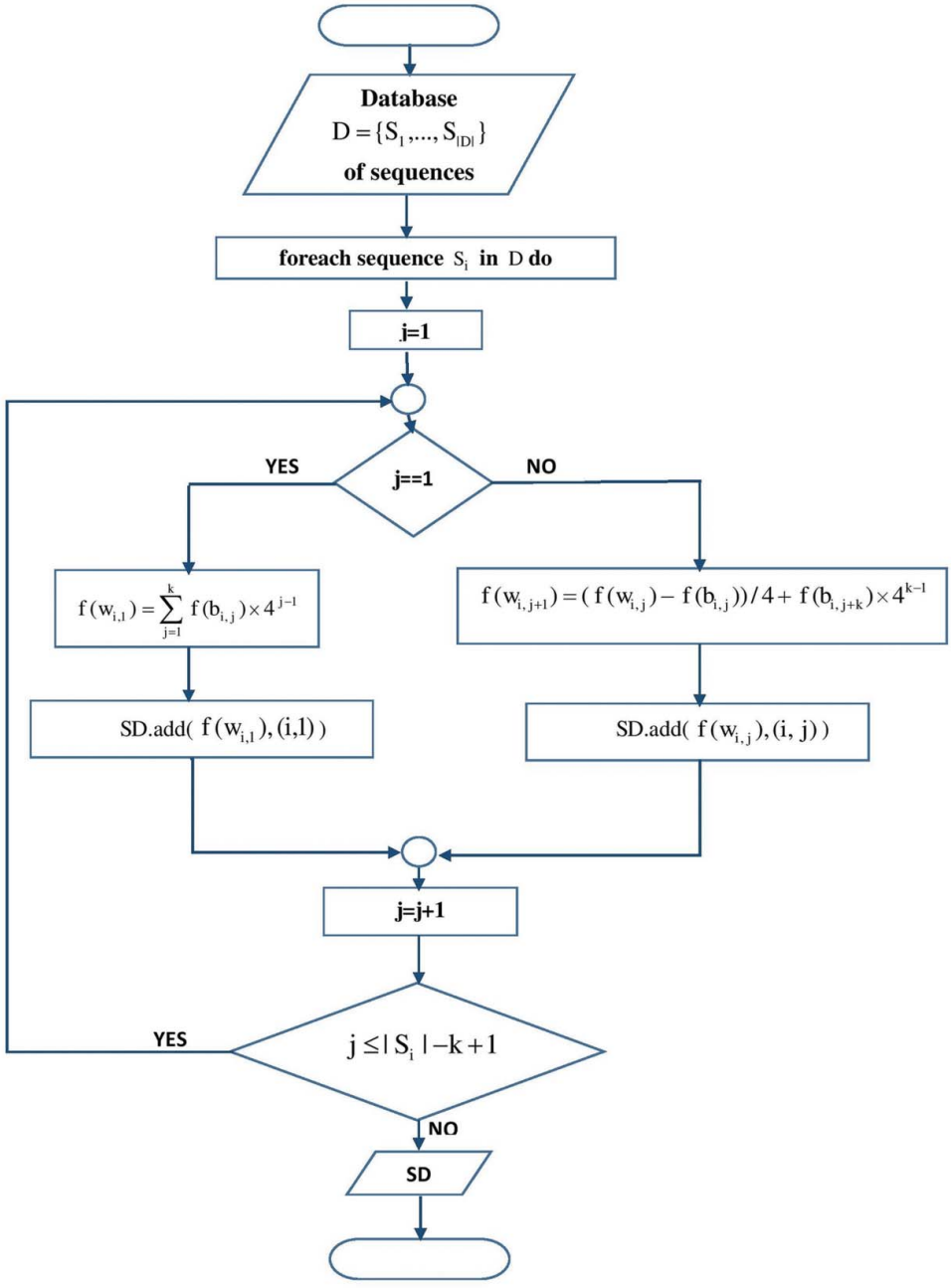

Figure 1. DNA data-indexing phase.

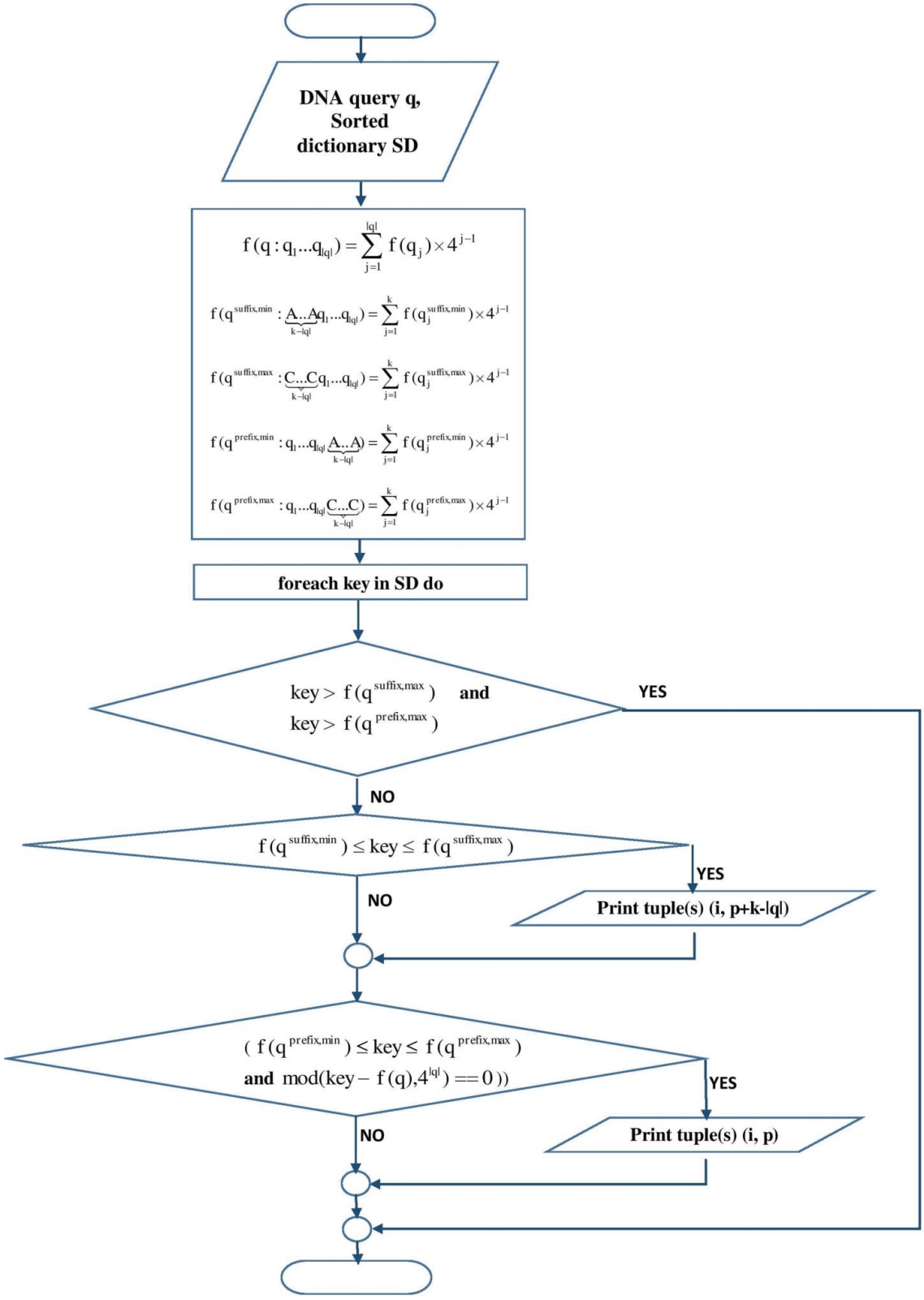

Figure 2. DNA pattern searching phase.

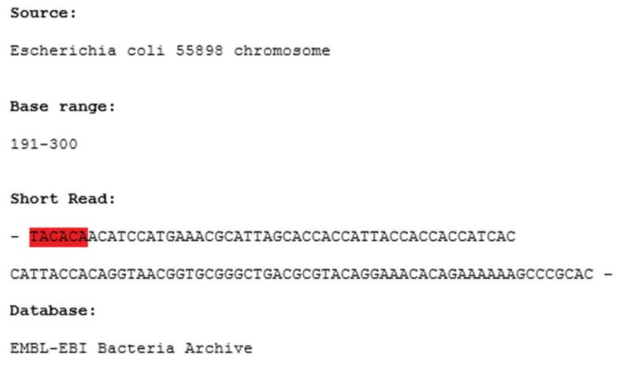

Figure 3. Short-read form *E. coli* 55989 chromosome, base range: 191−300.

Results and discussion

Let us consider the short-read form *E. coli 55989 chromosome* in the base range 191−300 (Figure 3). If $k = 8$, a search of the short read for *E. coli thrL* gene promoter consensus **TACACA**, which is located at the **−10** position relatively to the TSS (transcription start site), by the algorithm of Reneker and Shyu would not report the hit at the beginning of the short read. This is because when this algorithm is applied for a DNA pattern search shorter than k nucleotides, only hits that are found as suffixes of mapped words are reported. The consensus sequence **TACACA** is located at the start of the short read (Figure 3) and none of the mapped overlapping words of $k = 8$ nucleotides: **TACACAAC**, **ACACAACA**, **CACAACAT**..., contain the searched DNA pattern **TACACA** as a suffix. Therefore, the **TACACA** hit at position 0 in the short read is not reported by Reneker and Shyu's approach.

The improvement proposed by us incorporates not only a suffix search strategy, but also a prefix search strategy based on Equations (3) and (4) and, thus, the hit at the beginning of the short read is reported. Before performing any check, $f(q) = f(TACACA) = 2182$, $f(q^{\text{prefix},\,min}) = f(TACACAAA) = 22662$ and $f(q^{\text{prefix},max}) = f(TACACACC) = 84102$ are computed. When the search comes to the sorted dictionary $\boldsymbol{key} = 71814$, which corresponds to the mapped value of the word **TACACAAC**, and which contains the searched DNA pattern as a prefix, Equations (3) and (4) are satisfied: $22662 = f(q^{\text{prefix},min}) \leq key = 71814 \leq f(q^{\text{prefix},max}) = 84102$ and $\text{mod}(key - f(q), 4^{|q|}) = \text{mod}(71814 - 2182, 4^6) = \text{mod}(69632, 4096) = 0$. The consensus hit at the beginning of the short chromosome read is reported, represented with the tuple $(i, 0)$, given that the short read is the ith DNA sequence being indexed.

By applying the proposed computational upgrade of the indexing formula used in SSAHA and Reneker and Shyu's algorithm, the complete *E. coli 55989* chromosome, which contains 5 Mb (mega bases), retrieved from the European Nucleotide Archive was indexed for one minute. Excluding the computational upgrade, eight minutes were spent for the same purpose, given that all *E. coli 55989* overlapping words of $k = 8$ nucleotides were mapped. Since the time complexity of the proposed computational upgrade is linear and the entire *E. coli* DNA data-set of 0.1 Gb was indexed in 20 minutes, the whole human genome, containing approximately 3 Gb, could be indexed in 600 minutes (10 hours). Using the same computational resources, the straightforward application of the indexing formula proposed by Reneker and Shyu would require 3.3 days. When compared to Simpson and Durbin,[32] who estimated that 4.5 days and 700 GB RAM would be required in order to index the human genome, the proposed computational upgrade results in an 11-fold speed-up.

The storage improvement was also experimentally analysed. For instance, given that the length of the mapped words equals 8, SSAHA pre-computes a hash table with $4^8 = 65,536$ keys. If the indexed data structure is constructed dynamically as proposed, the number of records in the data structure increases with the size of the indexed DNA data (Table 1). Using the same computational resources, the number of records in the indexed data structure was determined for indexing of 1, 2, 3, 4 and 5 Mb, extracted from *E. coli 55989* chromosome (Table 1). According to the results obtained, 64,422 records were tracked in the indexed data structure for 1 Mb DNA. The number of records in the indexed data structure increased to 65,471 records, which were tracked when the entire *E. coli 55989* chromosome was mapped. Even in that case, SSAHA unnecessarily keeps $65,536 - 65,471 = 65$ keys, which map words of eight nucleotides that were not found in *E. coli 55989* chromosome. These 65 records are excluded from the indexed data structure, if the indexed data structure is constructed dynamically as we propose.

The use of a sorted dictionary instead of a hash table enables faster identification of all DNA pattern hits. In three out of the five cases, when searching for different promoter consensus sequences recognized by σ^{28}, σ^{54} and σ^{70} transcription factors (Table 2 and Figure 4), our

Table 1. Comparison of the number of records in the indexed data structures.

Base range (Mb)	SSAHA	Dynamic construction of the indexed data structure	Number of redundant records
1	65,536	64,422	1114
2	65,536	65,147	389
3	65,536	65,346	190
4	65,536	65,424	112
5	65,536	65,471	65

Table 2. Comparison of the running times for searching different promoter consensus sequences

Sigma factor	Consensus query	Conversion value	Reneker and Shyu (ms)	Our algorithm (ms)
σ^{28}	CCGATAT	f(CCGATAT) = 9860	9	6
σ^{70}	TATAAT	f(TATAAT) = 2406	8	8
σ^{70}	TTGACA	f(TTGACA) = 2170	10	8
σ^{54}	CTGGTA	f(CTGGTA) = 1788	13	13
σ^{28}	CTAAA	f(CTAAA) = 348	17	15

algorithm ran faster than the one of Reneker and Shyu. The improvement is due to the computational concept of the proposed searching approach, which is able to identify the same DNA pattern hits as the one of Reneker and Shyu, but without having to examine all records in the indexed data structure. It is also noteworthy that the higher the conversion value of the searched DNA pattern is, the less records are examined, resulting in better search time performance.

In addition to the computational improvements, the main conceptual improvement of the proposed methodology, in comparison to the method of Reneker and Shyu, is the ability for detection of all DNA pattern hits, regardless of their starting positions in the sequences. There are no reports until now that SSAHA and the algorithm of Reneker and Shyu are not able to detect DNA pattern hits shorter than k nucleotides, which are located at the beginnings of the sequences. This drawback can be solved by employing a prefix search that provides more exact matching in comparison to Reneker and Shyu's algorithm (Table 3).

According to the data in Table 3, the number of unidentified hits, when searching the indexed data structure for consensus elements **CCGATAT**, **TATAAT**, **TTGACA**, **CTGGTA** and **CTAAA**, ranges between 6 and 26. By applying the proposed methodology, a total of 65 unreported consensus hits located at the beginning of the indexed *E. coli* DNA fragments were additionally identified (Table 3).

Taken together, the results demonstrate that the improvements proposed by us give better *computational*, *storage* and *matching* performances in comparison to SSAHA and Reneker and Shyu's algorithm, using the same computational resources. To summarize the

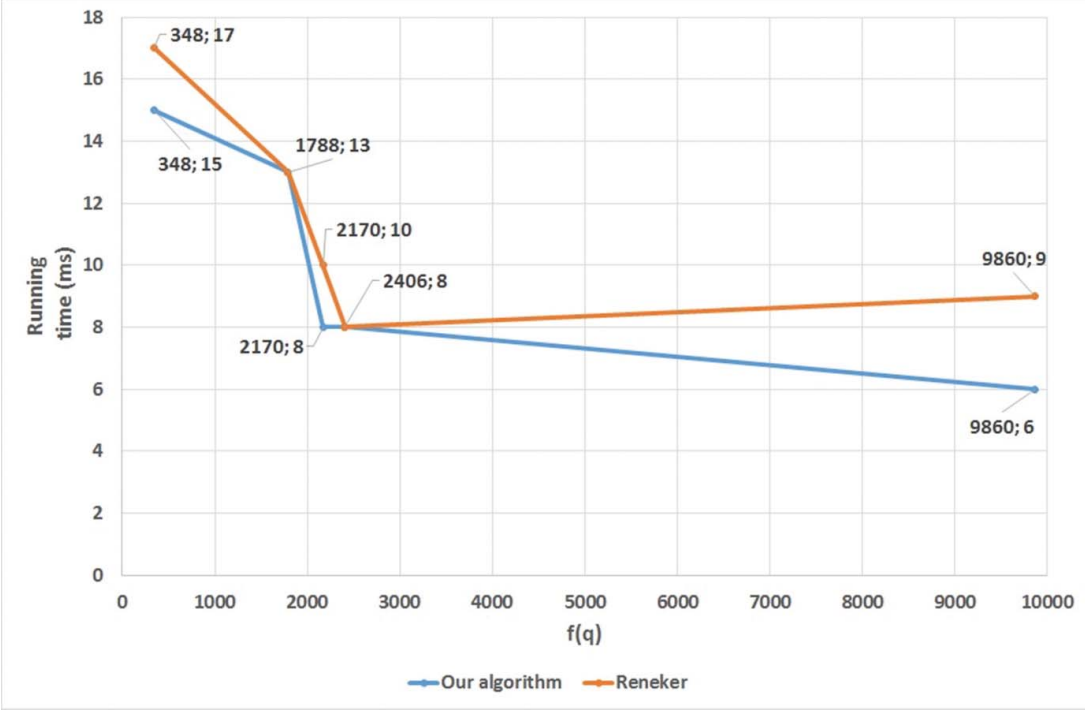

Figure 4. Comparison of the results in Table 2.

Table 3. Comparison of the matching rates

Sigma factor	Consensus query	Conversion value	Number of hits – Reneker and Shyu	Number of hits – our algorithm	Number of hits unreported by Reneker and Shyu
σ^{28}	CCGATAT	$f(\text{CCGATAT}) = 9860$	2520	2526	6
σ^{70}	TATAAT	$f(\text{TATAAT}) = 2406$	3812	3819	7
σ^{70}	TTGACA	$f(\text{TTGACA}) = 2170$	4010	4019	9
σ^{54}	CTGGTA	$f(\text{CTGGTA}) = 1788$	12,001	12,018	17
σ^{28}	CTAAA	$f(\text{CTAAA}) = 348$	18,140	18,166	26

Table 4. Summary of the advantages of the presented methodology

Factor	SSAHA	Reneker and Shyu	Our algorithm
Time complexity of the indexing phase	O(nk)	O(nk)	O(n) – k times faster
Structure of indexed data structure	Hash table with 4^k records	File	Sorted dictionary with less than 4^k records for a small DNA database
Matching rate	Reports limited number of hits	Better matching rate than SSAHA, but unable to detect hits at the beginnings of the sequences	Identifies all DNA pattern hits

obtained results, Table 4 lists the advantages of the methodology proposed by us in comparison to the other two algorithms.

The biological impact of the proposed methodology lies in the ability to detect all DNA pattern hits, regardless of their starting positions in the sequences. Unlike SSAHA and the approach of Renker and Shyu, our algorithm can identify DNA pattern hits which are located at the beginnings of the indexed DNA sequences, by integrating suffix search and prefix search. As a consequence, the algorithm proposed by us is expected to provide more precise matching in DNA pattern analyses by being able to identify key DNA pattern hits that might indicate functional DNA data unreported by SSAHA and Reneker and Shyu's method.

Conclusions

This work presents suggestions for computational and conceptual improvements of the most commonly used hash-based implementations for DNA database indexing and searching, such as SSAHA and Reneker and Shyu's improvements of the SSAHA software tool. We propose a computational upgrade of the indexing formula used in SSAHA and the algorithm of Reneker and Shyu, by which the database can be indexed k-times faster. This is of particular importance if large eukaryotic DNA sequences have to be tracked. Another improvement of our algorithm as compared to SSAHA is more efficient memory use when a relatively short DNA database is indexed: by dynamic construction of the indexed data structure, all redundant keys are excluded (i.e. those to which correspond hash values of words not found in the database), which improves the storage aspects. In addition, we use a sorted dictionary instead of a hash table as an indexed data structure, in order to be able to identify the same hits as Renker and Shyu's algorithm but without having to scan the entire data structure. As a result, the proposed methodology was demonstrated to run faster than Reneker and Shyu's algorithm. These better *computational*, *storage* and *matching* results as compared to SSAHA and Reneker and Shyu's approach, when using the same computational resources, indicate that our algorithm can be considered a promising improvement.

Funding

This project was supported by "University Goce Delčev", project 'Development of novel algorithms and software library for biomedical engineering application'.

References

[1] Kirilov KT, Golshani A, Ivanov IG. Termination codons and stop codon context in bacteria and mammalian mitochondria. Biotechnol Biotechnol Equip. 2013;27(4): 4018–4025.

[2] Kirilov K, Ivanov I. A programme for determination of codons and codons context frequency of occurrence in sequenced genomes. Biotechnol Biotechnol Equip. 2012;26(5):3310–3314.

[3] Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol. 1970;48(3):443−453.
[4] Smith T, Waterman M. Identification of common molecular subsequences. J Mol Biol. 1981;147(1):195−197.
[5] Lipman DJ, Pearson WR. Rapid and sensitive protein similarity searches. Science. 1985;227(4693):1435−1441.
[6] Altschul S, Gish W, Miller W, Myers E, Lipman D. Basic local alignment search tool. J Mol Biol. 1990;215 (3):403−410.
[7] Blast online tool [Internet]. Rockville Pike, Bethesda: National Library of Medicine; [cited 2014 Apr 20]. Available from: http://blast.ncbi.nlm.nih.gov/Blast.cgi.
[8] Stokes WA, Glick BS. MICA: desktop software for comprehensive searching of DNA databases. BMC Bioinform. 2006;7:427.
[9] Meek C, Patel JM, Kasetty S. OASIS: an online and accurate technique for local-alignment searches on biological sequences. In: Freytag JC, Lockemann PC, Abiteboul S, Carey MJ, Selinger PG, Heuer A, editors. Proceedings of 29th International Conference on Very Large Data Bases; 2003 Sep 9−12; Berlin: Elsevier Science & Technology; 2003.
[10] Delcher AL, Kasif S, Fleischmann RD, Peterson J, White O, Salzberg SL. Alignment of whole genomes. Nucleic Acids Res. 1999;27(11):2369−2376.
[11] Delcher AL, Phillippy A, Carlton J, Salzberg SL. Fast algorithms for large-scale genome alignment and comparison. Nucleic Acids Res. 2002;30:2478−2483.
[12] Kurtz S, Philippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, Salzberg SL. Versatile and open software for comparing large genomes. Genome Biol. 2004;5:R12.
[13] Abouelhoda MI, Kurtz S, Ohlebusch, E. Replacing suffix trees with enhanced suffix arrays. J Discrete Algorithms. 2004;2:53−86.
[14] Kurtz S. Reducing the space requirement of suffix trees. J Softw Pract Exp. 1999;29(13):1149−1171.
[15] Khan Z, Bloom JS, Kruglyak L, Singh M. A practical algorithm for finding maximal exact matches in large sequence datasets using sparse suffix arrays. Bioinformatics. 2009;25(13):1609−1616.
[16] Vyverman M, De Baets B, Fack V, Dawyndt P. EssaMEM: finding maximal exact matches using enhanced sparse suffix arrays. Bioinformatics. 2013;29(6):802−804.
[17] Ferragina P, Manzini G. Opportunistic data structures with applications. In: IEEE, editor. Proceedings of the 41st IEEE Symposium on Foundations of Computer Science; 2000 Nov 12−14; Redondo Beach: IEEE Computer Society; 2000.
[18] Grossi R, Vitter JS. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. SIAM J Comput. 2005;35(2):378−407.
[19] Burrow M, Wheeler DJ. A block sorting lossless data compression algorithm. Palo Alto (CA): Digital Equipment Corporation; 1994. (Technical Report 124).
[20] Lippert RA, Mobarry CM, Walenz BP. A space-efficient construction of the Burrows−Wheeler transform for genomic data. J Comput Biol. 2005;12(7):943−951.
[21] Lippert RA. Space-efficient whole genome comparisons with Burrows−Wheeler transforms. J Comput Biol. 2005;12(4):407−415.
[22] Lam TW, Sung WK, Tam SL, Wong CK, Yiu SM. Compressed indexing and local alignment of DNA. Bioinformatics. 2008;24(6):791−797.
[23] Ohlebusch E, Gog S, Kugell A. Computing matching statistics and maximal exact matches on compressed full-text indexes. In: Chávez E, Lonardi S, editors. Proceedings of the 17th Annual Symposium on String Processing and Information Retrieval; 2010 Oct 11−13; Los Cabos: Springer; 2010.
[24] Benson G. Tandem repeats finder: a program to analyze DNA sequences. Nucleic Acids Res. 1999; 27(2):573−580.
[25] Rigoutsos I, Floratos A. Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm. Bioinformatics. 1998;14(1):55−67.
[26] Miller C, Gurd J, Brass A. A RAPID algorithm for sequence database comparisons: application to the identification of vector contamination in the EMBL databases. Bioinformatics. 1999;15(2):111−121.
[27] Giladi E, Walker MG, Wang JZ, Volkmuth W. SST: an algorithm for finding near-exact sequence matches in time proportional to the logarithm of the database size. Bioinformatics. 2002;18(6):873−877.
[28] Ning Z, Cox AJ, Mullikin JC. SSAHA: a fast search method for large DNA databases. Genome Res. 2001;11 (10):1725−1729.
[29] Reneker J, Shyu C-R. Refined repetitive sequence searches utilizing a fast hash function and cross species information retrievals. BMC Bioinform. 2005;6(1):111.
[30] Kent WJ. BLAT − the BLAST-like alignment tool. Genome Res. 2002;12(4):656−664.
[31] The European Nucleotide Archive [Internet]. Heidelberg: The European Molecular Biology; [cited 2014 Apr 20]. Available from: http://www.ebi.ac.uk/ena/.
[32] Simpson JT, Durbin R. Efficient construction of an assembly string graph using the FM-index. Bioinformatics. 2010;26(12):i367−i373.