Advances in Robotics & Automation
ISSN: 2168-9695

Law and Cheng, Adv Robot Autom 2014, 3:1
DOI: 10.4172/2168-9695.1000118

Review Article Open Access

A Survey of Techniques for Sequence Similarities Matching in Compression

Law NF* and Cheng KO

Center for Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Hong Kong

Abstract

DNA Sequence Compression can be achieved through exploiting the intra-sequence and inter-sequence similarities. In order to have a good compression gain, effective methods have to be used to search for all the similar subsequences within the DNA sequences so that these similar subsequences can be encoded together. Different pattern recognition methods have been used to search for these similar subsequences. They are suffix-based approach, dynamic programming approach, seed extension approach, rule-based approach and parsing approach. A survey of their main ideas and application domains has been given in this paper.

Keywords: DNA Sequence Compression, suffix-based approach, dynamic programming approach, seed extension approach, rule-based approach and parsing approach

Introduction

Study of DNA sequences is important in areas such as forensic application and crime investigation. Due to the advancement in DNA sequencing technologies, the number of DNA sequences stored in public databases has been increased significantly in recent years [1,2]. Typically, these DNA sequences in public databases are in "uncompressed form". The uncompressed data not only need large storage, but also make sequence distribution ineffective. As a result, many lossless compression algorithms have been proposed for compressing the DNA sequences.

Traditionally, DNA sequences are compressed by exploring intra-sequence similarity (also known as the horizontal mode [3]). In other words, identical subsequences within the DNA sequence to be compressed are searched and encoded together to achieve compression. As DNA sequence often does not have strong intra-sequence similarity, these intra-sequence based compression methods cannot give a high compression gain. An average of less than 10% compression gain is obtained for most DNA sequences.

In order to achieve a better compression, DNA sequence compression method uses inter-sequence similarity (also known as the vertical mode [3]) in addition to intra-sequence similarity. In other words, a reference DNA sequence or a set of reference sequences is considered. Similar subsequences within the set of the reference sequences and the DNA sequence to be compressed are searched and encoded together. The inter-sequence based compression method is very effective for compressing a set of genome-wide sequences. A compression gain of over 1000% can sometimes be achieved if proper reference sequences are used in searching the inter-sequence similarities.

For both intra-sequence and inter-sequence compression methods, a crucial component is the identification of similar subsequences. A higher compression is achieved if more similar subsequences are found. Many different pattern recognition methods have been used for identifying the similar subsequences. They include suffix-based, dynamic programming, seed extension, rule-based and parsing methods. In this paper, a survey of these different techniques is provided while other reviews of DNA sequence compression with different focuses can be found in [4-8].

This paper is organized as follows. Existing DNA Sequences Compression Algorithms first describes the basic principles of DNA sequences compression algorithms. Techniques for Searching Similar Subsequences then discusses the different pattern recognition techniques used for searching similar patterns. In Discussion and Future Development these techniques are briefly compared. A discussion on future development is also given. Finally, Conclusion concludes the paper.

Existing DNA Sequences Compression Algorithms

A DNA sequence is a long sequence consisting of four types of bases: Adenine (A), Cytosine (C), Guanine (G) and Thymine (T). As DNA sequence has a double helix structure, nucleotide in one DNA strand would bind to its complementary nucleotide in the other strand. The nucleotide A is complementary to T while C is complementary to G. Knowledge of one strand can loss lessly reconstruct another strand. Hence, compressing a DNA sequence means compressing one strand in the double helix DNA structure. Without compression, two bits per base are required for encoding the four nucleotides {A, C, G, T}. A DNA sequence compression algorithm thus needs to achieve a bit per base smaller than two in a lossless manner.

There are generally two classes of methods for DNA sequences compression. They are intra-sequence compression method and inter-sequence compression method. The intra-sequence compression is based on the fact that subsequences which are thousands of bases apart in a DNA sequence could have similar bases compositions and orderings. Figure 1 shows a conceptual idea of intra-sequence similarity. The DNA sequence S consists of six subsequences {A, B, C, D, E, F}. Bases in subsequence B are identical to those in F. Similarly,

***Corresponding author:** Law NF, Center for Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Hong Kong, Tel: 27664746; E-mail: ennflaw@polyu.edu.hk

Received March 28, 2014; **Accepted** April 16, 2014; **Published** April 18, 2014

Citation: Law NF, Cheng KO (2014) A Survey of Techniques for Sequence Similarities Matching in Compression. Adv Robot Autom 3: 118. doi: 10.4172/2168-9695.1000118

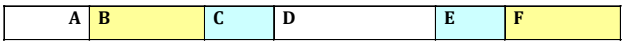

Figure 1: A DNA Sequence, *S*.

bases in C and E are also identical. Then the intra-sequence similarity implies that E and F can be obtained through C and B with proper

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
base	T	T	A	G	C	G	T	T	A	G	C	G	C	G	C	T	A	A

indexing information. Hence, the compression algorithm only needs to compress the subsequences {A, B, C, D} which results in a reduction of the compression size.

The similar subsequences can be exact repeats with the same bases or with reverse complementary bases [9, 10]. Consider the sequence S_1 as,

The sequence S_1 contains both repeat and reverse complementary repeat. Subsequence from base 1 to base 6 and subsequence from base 7 to base 12 are repeats (both are TTAGCG). The subsequence "CGCTAA" in the last six bases is reverse complementary repeat of the subsequence of the first six bases (TTAGCG). This is because the complementary bases of "CGCTAA" are "GCGATT". The

Position	1	2	3	4	5	6	7	8	9
base	T	T	A	G	C	T	A	A	C

reverse ordering of "GCGATT" is "TTAGCG" which is same as the subsequence in the first six bases. Hence encoding S1 reduces to compress the first six bases "TTAGCG" together with some indexing information for reconstructing bases from position 7 to18.

Besides exact repeats, approximate repeats can also be considered [11-15]. The approximate repeats mean that the two subsequences have essentially the same bases but may have some mismatched bases. The mismatched bases can be obtained through substitution, deletion and insertion operations. Consider the sequence S_2 as,

The subsequence from base 6 to 9 can be obtained from the subsequence from base 1 to 5 by substitution and deletion operations. In particular, "TAAC" is obtained by replacing the second base in "TTAGC" by A and removing the fourth base "G". Encoding S_2 reduces to compress the first five bases together with some indexing information and edit operations for reconstructing bases from position 6 to 9. Examples of intra-sequence DNA compression methods include BioCompress [3], Cfact [16], GenCompress [11-13], DNACompress [14], DNAPack [15], DNAC [17] and GeNML [18,19].

The inter-sequence compression methods also consider finding similar subsequences. In contrast to intra-sequence compression, inter-sequence compression finds similar subsequence from other DNA sequences called reference sequences. Figure 2 shows a conceptual diagram depicting the inter-sequence similarity between the DNA sequence S and its reference sequence *R*. The following subsequences are assumed to be repeats: {B, F, H}, {C, E} and {D, G}. Hence, by using proper indexing and assuming the reference sequence R is available, we only need to compress the sequence *S* as {A, C}, instead of {A, B, C, D} as in Figure 1. As opposed to the consideration of intra-sequence similarity only, the use of inter-sequence similarity always results in a better compression gain.

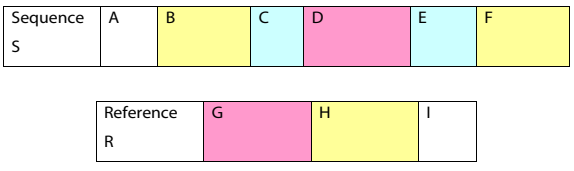

Figure 2: A DNA Sequence, S and the reference sequence R.

An important issue in inter-sequence based compression methods is the choice of the reference sequence. If the reference sequence has a high similarity to the DNA sequence to be compressed such as the DNA sequences from different individuals of the same species, a high compression gain can be obtained. Examples of popular inter-sequence DNA compression methods include the delta compression method that encoded the base-to-base differences, [20-24], the Lempel-Ziv (RLZ) compression [25,26], RLCSA [27], COMRAD [28] and GDC [29,30].

Irrespective of which compression method is used, an important component of the two classes of methods is the searching of similar subsequences. If many long similar subsequences are obtained, a high compression gain can be achieved. Many methods have been used in finding similar subsequences in DNA compression methods. These methods use different pattern recognition methods to identify similar subsequences. In the next section, a survey of these pattern recognition methods is given.

Techniques for Searching Similar Subsequences

An important component of the DNA sequence compression methods is the searching of similar subsequences. Pattern recognition methods are often used to identify these similar subsequences. In this section, methods for searching similar subsequences are classified into five groups; namely suffix-based, dynamic programming, seed extension, rule-based and parsing approaches.

Suffix-based approaches

Suffix-based approaches such as suffix tree and suffix array have been used in both intra-sequence and inter-sequence DNA compression. A suffix of a sequence *S* can be represented as S[*i, n*], where *n* is the length of the sequence *S* and *i* is between 1 and n. If a subsequence is repeated in *S*, the repeated subsequence would appear as a common prefix in certain suffixes *S*[*i, n*]. The suffix tree representation [31] is usually applied to a sequence S ended with an additional symbol $ which denotes the termination of a sequence, i.e. *S*$. The *i*th terminal node corresponds to the *i*th suffix S[*i, n*]. Each edge is labeled with a substring in *S*$ such that the *i*th suffix can be obtained by traversing from the root to the *i*th terminal node and concatenating the substrings in the edges. Meanwhile, each nonterminal node branches to at least two children and acts as the divergence point for some suffices after the longest common prefix.

An example of a suffix tree for the sequence ATATGTA$ is shown in Figure 3. Starting from the root node, all subsequences can be obtained by traversing the suffix tree. For example, S[1, *n*]=ATATGTA$ while S[3, *n*]=ATGTA$. Comparing the traversing of S[1, *n*] and S[3, n], the repeated sequence "AT" appears as their common prefix. Another example is S[2, *n*] (TATGTA$) and S[6, *n*] (TA$). The repeated sequence "TA" appears as the common prefix. In the intra-sequence based algorithm Cfact [16], the suffix tree is employed to efficiently

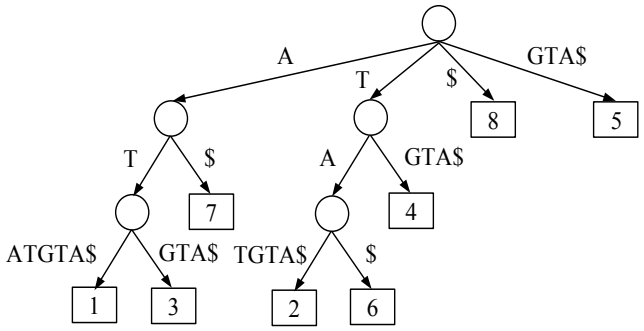

Figure 3: A suffix tree of a sequence ATATGTA$.

identify a similar subsequence for the currently un encoded portion of the sequence as the longest common prefix.

The inter-sequence based algorithm, run-length compressed suffix array (RLCSA) [27] considers the other structure called suffix array which represents a sorted array of suffixes in lexicographic order. The *i*th element in a suffix array, $A[i]$, is an index of a sequence S$ such that S [A[i], n+1] < $S[A[i+1], n+1]$ for . The inequality is defined such that the symbol $ is always less than all the other symbols while $S[A[i], n+1]$ is ordered before $S[A[i+1], n+1]$ in the lexicographic order. The repeated patterns are localized in the suffix array and so does their sub-patterns. To remove the redundancy, RLCSA first transforms the suffix array $A[i]$ into another sequence $\psi(i)$ which satisfies the condition $A[\psi[i]]=A[i]+1$. Then, the difference $\psi[i+1]-\psi[i]$ is calculated and encoded using run-length coding for compression. The compressed data structure can be extended for inter-sequence compression by concatenating a set of sequences together. Furthermore, RLCSA has an added advantage of allowing random sequence access. This is achieved by including properly sampled indexing information.

Dynamic programming approaches

Dynamic programming has been used for finding similar subsequences in different sequences including DNA sequences [15, 32]. In intra-sequence based compression algorithms such as DNA Pack [15], the selection of similar subsequences is formulated as an optimization problem which is solved recursively through dynamic programming. The compressed size is usually considered as the cost function for minimization in DNA sequence compression. Define $S[1, i]$ be the prefix of an input sequence S up to position i and C_0(i) be the minimal cost for compressing S[1, i]. Suppose that only substitutions are considered in similar subsequences. The minimal compression cost of the sequence S, i.e. $C_0(n)$, can be determined recursively from i = 2 using the following formula

$$C_0(i)=\min\{C_1(i),C_2(i),C_3(i)\} \quad (1)$$

Where $C_1(i),C_2(i)$ and $C_3(i)$ are minimal compression cost of S[1, i] by compressing the suffix of S[1, i] in different manners and are determined partially on $C_0(j)$ for $1\le j<i$. In particular, $C_1(i)$ considers a similar subsequence with a copy of no reverse complement in the way that

$$C_1(i)=\min_{1\le j<i,1\le k\le j}\{C_0(j)+C_r(\mathrm{j+1,i,k})\} \quad (2)$$

Where C_r(j+1,i,k) is the compression cost of the subsequence from j + 1 to i with reference to another subsequence of the same length but starting before it at $k(1\le \mathrm{k}\le \mathrm{j})$.$C_2$(i) involves a similar subsequence with a reverse complement copy and is given by

$$C_2(i)=\min_{1\le j<i,1\le k\le j}\{C_0(j)+C_p(\mathrm{j+1,i,k})\} \quad (3)$$

where C_p(j+1,i,k) is the compression cost of the subsequence from j + 1 to i with reference to a reverse complementary copy of a subsequence of the same length starting at is computed by treating the suffix as a subsequence of no similarity with the other as follows

$$C_3(i)=\min_{1\le j<i}\{C_0(j)+C_b(j+1,i)\} \quad (4)$$

where $C_b(j+1,i)$ is the cost using compression for non-repeated pattern such as, arithmetic coding [33].Initially, $C_0(1)=C_b(1,1)$ as there is no other selection. At the end, an optimal set of similar subsequences will be obtained by tracing back the recursive process. The dynamic programming has a high computation cost. Its direct implementation has a complexity of $O(n^3)$

Seed extension approaches

Although dynamic programming can identify optimal similar subsequences for a given cost function, its computational cost is high particularly for long DNA sequences. The cost is still too high even only approximate subsequences with substitutions are considered. Various seed-based approaches have been proposed to improve the computational complexity. For example, BLAST [34] improves the processing time by first searching for those short seed matches, i.e., the subsequence pairs at a small fixed length of k (e.g. k = 4) with a similarity score not less than a certain threshold. After identifying these short seed matches, the seed matches are extended to longer subsequence matches with another pre-set similarity score threshold.

Pattern Hunter [35] improves the sensitivity of BLAST by relaxing the constraint of seeds to have matches in non-consecutive positions. For example, a seed match of length k = 6 can be found using a mask 111011, where the value of 1 indicates a position required to be identical while 0 refers to a position which can be matched or not matched (a state of "don't cares").The intra-sequence compression algorithm DNA Compress [14] adopts Pattern Hunter to search subsequence matches for compressing a single DNA sequence. As the subsequence matches may overlap with each other, DNA Compress applies a greedy strategy to remove the overlap such that the subsequence matches with a larger similarity score are preserved in the overlapping region.

In contrast to the dynamic programming approach, the seed extension approach has been exploited for multiple sequence compression in [9,10,29,30]. In [9,10], a set of chromosome sequences is compressed together in which identical subsequences are searched within all the chromosome sequences in addition to itself. Another algorithm called GDC [29,30] applies hashing to a reference sequence so that seed matches can be searched from the reference sequences for other sequences. A heuristic strategy is proposed to extend the seed matches for minimizing the compressed size. In addition, random access can be enabled by dividing each sequence into blocks of nearly equal size. However, the block strategy makes no similar subsequences found across block boundaries.

Rule-based approaches

The rule-based compression approaches usually involve grammar rules which can capture long term repetitions in data. A grammar rule can be expressed in the form $X \rightarrow r$, where r is a subsequence in a sequence S and X is a symbol which can represent and substitute r. For

example, a DNA sequence S = CGATGACGAT can be represented by YXY using two rules X→GA and Y→CXT. As seen, a rule can be created to describe a repeated pattern. The rule-based approaches have been applied in both intra-sequence and inter-sequence compression. DNA Sequitur [36] applies rules designed for exact subsequence matches and those of reverse complements in intra-sequence compression. The algorithm alternates between construction of rules and generation of sequence representation using the new rules until no more rule is generated. Each rule is constructed for a repeated pair of symbols, any of which may be one of the original symbols in the DNA sequences or the symbols substituted previously using other rules. The final representation of the sequence and the grammar rules are entropy-encoded to obtain a compressed bitstream.

The inter-sequence compression method COMRAD [28] applies the rule-based principle for capturing the inter-sequence similarity. However, only subsequence matches which satisfy certain frequency and pattern requirements are kept in the rules, due to the consideration of the compressed size and computational complexity. Besides, random access can be realized by including extra information about locations of subsequence matches described by the rules.

Parsing approaches

The parsing approaches can be found in the intra-sequence compression method such as LZ77 [37] and the inter-sequence compression method such as RLZ [25]. In the parsing approaches based on intra-sequence similarity, a long sequence is usually divided into a number of separated phrases using a greedy strategy in which each of the phrases is the longest subsequence that has an identical copy found before it (if it does not begin with symbols not seen before). For example, a DNA sequence S = CGATGACGAT can be decomposed into phrases C, G, A, T, GA and CGAT. Reverse complement copies can also be considered to minimize compressed size for DNA sequences. RLZ [25] applies the parsing strategy by using a reference sequence as the searching space for identical copies of phrases.

The greedy search of subsequence matches is sub-optimal as implied by equations (1) – (4) used in dynamic programming. RLZ-opt [26] changes the parsing to be non-greedy by allowing lookahead after the parsing point. In other words, the longest match could be anywhere (in a restricted region) after the starting point of the unencoded portion of the sequence. This makes a potential longer subsequence matches for better compression.

Discussion and Future Development

In previous section, the pattern recognition methods for DNA sequence compression have been divided into five categories; suffix-based, dynamic programming, seed extension, rule-based and parsing approaches. Dynamic programming can search for optimal set of similar subsequences for compression. Despite that its high computational cost hindered its application in a large dataset such as individual sequences of the same population which becomes prevalent due to advanced sequencing technology. Seed extension methods compromise computational time and optimal compressed size by considering short seed matches. The methods are usually accompanied with some heuristic methods to extend the seed matches for longer similar subsequences.

The parsing approaches focus on searching identical copies for subsequences at current encoding point or lookahead in some restricted region. To further exploit the redundancy due to approximate similar subsequences, special encoding strategies such as finding longest increasing subsequence [26] are required. The suffix-based approach, suffix-tree, can be used as an efficient tool for finding identical subsequences but they do not provide efficient representation unless transformed into other structures such as RLCSA. Besides compression, RLCSA has an additional advantage of random access. However, even with RLCSA, experimental results in [25,26,30] show that the compression algorithms based on explicit identification of similar sequences such as RLZ and GDC can outperform RLCSA, likely because of the efficient representation based on LZ77.

The rule-based approaches provide alternative for describing similar subsequences. However, efficiently encoding of the dictionary produced by the grammar rules is also important in compression performance. DNA Sequitur may result in more than 2 bases per bit for intra-sequence compression [36]. Although another rule-based method COMRAD can achieve high compression ratio in inter-sequence compression, further improvement is required as compared with the parsing approach and the seed-based approaches when compressing individual sequences from the same population [26, 30].

While the use of different pattern recognition methods in searching similar subsequences would affect the compression performance, another important issue is the choice of the reference sequences. The reference sequence needs to be a good representation of the DNA sequences to be compressed. The use of a good reference sequence can significantly reduce the size of the compressed sequence [21,25]. There are different proposals for choosing the reference sequences. For example, in [29,30], the reference sequence is chosen as the one with the largest number of repeated patterns. Artificial reference sequences can also be considered. In [21], the consensus sequence was chosen as the artificial reference sequence. In [29,30], artificial reference sequence was constructed from a real sequence appended with extra phrases. In [38], the reference sequence was constructed by using the dictionaries produced from three different compression algorithms.

Despite the above investigation, no optimal method is obtained yet. As low bit per base is obtained primarily through a set of highly similar reference sequences, a reliable and systematic way to choose them is needed. Further research results confirm that the use of more than one sequence can reduce the compressed size as well [30]. However, the selection of an appropriate set of reference sequences could be challenging. This will be our future research direction.

Conclusion

State-of-the-art DNA sequences compression methods work by exploiting intra-sequence and/or inter-sequence similarities. Pattern recognition techniques are often used to search for similar subsequences in the DNA sequences. In this paper, a survey of different pattern recognition methods for identifying the similar patterns is performed. They include the suffix-based, dynamic programming, seed extension, rule-based and parsing approaches.

The suffix-based approach is a type of data structure that facilitates the searching of the longest similar subsequences in the data. The dynamic programming approach formulates the problem as an optimization problem which is then solved recursively. The seed extension method performs the searching by extending the size of the similar subsequence matches in multiple passes. The rule-based method models the similar subsequence as some kinds of rules. The parsing approaches divide the long sequence into different segments in which the segments are similar subsequences.

Besides compressed size, the pattern recognition methods have

impact on other issues such as random access. On the other hand, existing works show that the reference sequence is one of the critical factors in compression performance. In future, we will investigate the further improvement by selecting one or more appropriate reference sequences.

References

1. Bgenson DA, Cavanaugh M, Clark K, Karsch-Mizrachi I, Lipman DJ, et al. (2013) GenBank. Nucleic Acids Research 41: 36-42
2. Kodama Y, Shumway M, Leinonen R (2012) The sequence read archive: explosive growth of sequencing data. Nucleic Acids Research 40: 54-56.
3. Grumbach S, Tahi F (1994) A new challenge for compression algorithms: genetic sequences. Journal of Information Processing and Management 30: 875-886.
4. Giancarlo R, Scaturro D, Utro F (2009) Textual data compression in computational biology: a synopsis. Bioinformatics 25: 1575-1586.
5. Giancarlo R, Scaturro D, Utro F (2012) Textual data compression in computational biology: Algorithmic techniques. Computer Science Review 6: 1-25.
6. Bakr NS, Sharawi AA (2013) DNA lossless compression algorithms: review. American Journal of Bioinformatics Research 3: 72-81.
7. Zhu Z, Zhang Y, Ji Z, He S, Yang X (2013) High-throughput DNA sequence data compression. Briefings in Bioinformatics.
8. Giancarlo R, Scaturro D, Utro F (2013) Compressive biological sequence analysis and archival in the era of high-throughput sequencing technologies. Briefings in Bioinformatics.
9. Paula CPW (2009) An approach to multiple DNA sequences compression. MPhil Thesis, The Hong Kong Polytechnic University.
10. Wu P, Law NF, Siu WC (2009) Analysis of cross sequence similarities for multiple DNA sequences compression. International Journal of Computer Aided Engineering and Technology 1: 437-454
11. Chen X, Kwong S, Li M (2001) A compression algorithm for DNA sequences. IEEE Engineering in Medicine and Biology Magazine 20: 61-66.
12. Chen X, Kwong S, Li M (1999) A compression algorithm for DNA sequences and its applications in genome comparison. Genome Informatics 10: 51-61.
13. Li M, Badger JH, Chen X, Kwong S, Kearney P, et al. (2001) An information-based sequences distance and its application to whole mitochondrial genome phylogeny. Bioinformatics 17: 149-154
14. Chen X, Li M, Ma B, Tromp J (2002) DNA Compress: fast and effective DNA sequence compression. Bioinformatics 18: 1696-1698.
15. Behzadi B, Fessant FL (2005) DNA compression challenge revisited: a dynamic programming approach. Combinatorial Pattern Matching.Lecture Notes in Computer Science 3537: 190-200.
16. Rivals E, Delahaye JP, Dauchet M, Delgrange O (1996) A guaranteed compression scheme for repetitive DNA sequences. Processing of Data Compression Conference.
17. Chang CH (2004) DNAC: a compression algorithm for DNA sequences by non-overlapping approximate repeats. Master Thesis.
18. Tabus I, Korodi G, Rissanen J (2003) DNA sequence compression using the normalized maximum likelihood model for discrete regression. Proceedings of Data Compression Conference.
19. Korodi G, Tabus I (2005) An efficient normalized maximum likelihood algorithm for DNA sequence compression. ACM Transactions on Information System 23: 3-34.
20. Scott C, Yiming L, Chen L, Xiaohui X (2009) Human genomes as email attachments. Bioinformatics 25: 274-275.
21. Brandon MC, Wallace DC, Baldi P (2009) Data structures and compression algorithms for genomic sequence data. Bioinformatics 25: 1731-1738.
22. Afify H, Islam M, Wahed MA (2011) DNA lossless differential compression algorithm based on similarity of genomic sequence database. International Journal of Computer Science and Information Technology 3: 145-154.
23. Wang C, Zhang D (2011) A novel compression tool for efficient storage of genome resequencing data. Nucleic Acids Research.
24. Deorowicz S, Danek A, Grabowski S (2013) Genome compression: a novel approach for large collections. Bioinformatics 29: 2572-2578.
25. Kuruppu S, Puglisi SJ, Zobel J (2010) Relative Lempel-Ziv compression of genomes for large-scale storage and retrieval. String Processing and Information Retrieval, Lectures Notes in Computer Science 6393: 201-206.
26. Kuruppu S, Puglisi SJ, Zobel J (2011) Optimized relative Lempel-Ziv compression of genomes. Proceedings of the Thirty-fourth Australasian Computer Science Conference 113: 91-98.
27. Makinen V, Navarro G, Siren J, Valimaki N (2010) Storage and retrieval of highly repetitive sequence collections. Journal of Computational Biology 17: 281-308.
28. Kuruppu S, Beresford-Smith B, Conway T, Zobel J (2011) Iterative dictionary construction for compression of large DNA data sets. IEEE Transactions on Computational Biology and Bioinformatics 9: 137-149.
29. Grabowski S, Deorowicz S (2011) Engineering relative compression of genomes.
30. Deorowicz S, Grabowski S (2011) Robust relative compression of genomes with random access. Bioinformatics 27: 2979-2986.
31. B. Smyth, Computing Patterns in Strings, Pearson Education Limited, 2003.
32. Smith TF, Waterman MS (1981) Identification of common molecular subsequences. Journal of Molecular Biology 147: 195-197
33. Moffat A (1998) Arithmetic coding revisited. ACM Transactions on Information Systems 16: 256 – 294.
34. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. Journal of Molecular Biology 215: 403-410.
35. Ma B, Tromp J, Li M (2002) Pattern Hunter: faster and more sensitive homology search. Bioinformatics 18: 440-445.
36. Cherniavsky N, Ladner R (2004) Grammar-based compression of DNA sequences. UW CSE Technical Report 2007-05-02.
37. Ziv J, Lempel A (1977) A universal algorithm for sequential data compression. IEEE Transactions on Information Theory 23: 337 – 343.
38. Kuruppu A, Puglisi SJ, Zobel J (2011) Reference sequence construction for relative compression of genomes. String Processing and Information Retrieval, Lecture Notes in Computer Science 7024: 420-425.