Operational Research in Engineering Sciences: Theory and Applications
Vol. 6, Issue 1, 2023, pp. 322-331
ISSN: 2620-1607
eISSN: 2620-1747
DOI: https://doi.org/10.31181/oresta/0601131

COMPARATIVE ANALYSIS OF BIT-PARALLEL STRING PATTERN MATCHING ALGORITHMS FOR BIOLOGICAL SEQUENCES

Muhammad Yusuf Muhammad[1*], Mathias Fonkam[1], Salu George Thandekatu[1], Sandip Rakshit[1], Rao Narasimha Vajjhala[1]

[1]American University of Nigeria, PMB 2250 Yola, Adamawa State, Nigeria.

Received: 17 January 2023
Accepted: 26 March 2023
First Online: 15 April 2023

Research Paper

Abstract: *The inherent parallelism in a bit operation like AND/OR inside a computer word is known as bit parallelism. It plays a greater role in string pattern matching and has good application in the analysis of biological data. The use of recently developed bit parallel string matching algorithms approaches help in improving the efficiency of the other string pattern matching algorithms. This paper discusses the working of some of these bit parallel string matching algorithms and their application on biological sequences. It also shows how bit-parallelism can be efficiently used to address various matching problems in Bioinformatics to analyze biological sequences such as Deoxyribonucleic acid (DNA), Ribonucleic acid (RNA) and Protein with examples. It can also serve as greater tool for the researchers when looking for the appropriate method to us on Biological sequences.*

Keywords: *Bit-parallelism, Automaton, Pattern matching, Ribonucleic acid, Parameterized matching.*

[*] Corresponding author: Muhammad.muhammad@aun.edu.ng (M. Y. Muhammad) m.fonkam@aun.edu.ng (M. Fonkam), George.thandekkattu@aun.edu.ng (S. G. Thandekatu), Sandip.rakshit@aun.edu.ng. (S. Rakshit), narasimharao@gmail.com. (R. N. Vajjhala)

1. Introduction

String pattern is one of the important components of Bioinformatics that plays a greater role in the analysis of biological sequences. It is a technique that finds copies of one or more string pattern(s) (query string) from larger string text (Reference string). This method has been a greater tool used by computer science researchers in analyzing biological sequences such as finding homology between the query sequence (uncharacterized) and larger sequences (Fully characterized) in a sequence-database that help in inferring the function of newly discovered biological sequence this is because, homologous sequences tends to haves similar structure hence similar functions (Amir & Nor, 2007). Since these biological sequences are presented in the form of strings sequence i.e. DNA is presented as a string of four alphabets {A, C, G, T} each representing the first letter of each nucleotide base in DNA molecule (A for adenine, G for guanine, C for cytosine, T for thymine), RNA also is presented as a string of alphabets {A, C, G, U} as in DNA but RNA has Uracil (U) base instead of thymine and Protein is presented as string of twenty (20) alphabets representing the 20 amino acids in the protein molecule, string pattern matching approach tend to be most suitable in solving the problems of biological sequence analysis. Therefore, the problem of string pattern matching on biological sequence can be formally defined as:

Definition: Let T be a set of a large sting biological sequence [t1...tn] and P be a set of a string patterns [p1…pm] where n and m are respectively length of T and P. Then, the process is to locate all exact or approximate occurrences of a string pattern P in T (Allali & Sagot, 2005; Baker, 1995).

Example*,* given two disjoint finite sets of alphabets $\sum$ and $\prod$, the problem is to for all ti $\in \sum$ that exactly or approximately matched a pattern p $\in \prod$; where $\sum$ and $\prod$ may be DNA, RNA or Protein alphabets.

Myriads of string pattern matching algorithms for biological sequences analysis. For example, method for matching RNA secondary structure by finding the edit distance between bases and their complementary pairs was proposed in Cantone and Faro (Cantone & Faro, 2014). Another algorithm that finds the exact matching of RNA secondary structure expression was proposed in Miao, Chang, and Wang (2010) in O(nm2) time complexity (Miao et al., 2010). An indexing approaches discussed in Ďurian et al. (2010); Faro et al. (2012) uses a bidirectional affix tree data structure for exact and inexact pattern matching of RNA structure in O(nm) and O(n) worst- and best-case time (Ďurian et al., 2010; Faro et al., 2012). A more efficient data structures called affix array with affix tree functionality and structural suffix tree was used for RNA based pattern matching was studied in Grabowski and Fredriksson (Grabowski & Fredriksson, 2008). Hybrid approach for determining the function of a protein by comparing Protein structure using Suffix Array and Wavelet (PSISAW) was proposed in (Gharib et al., 2008). The Bit- parallel approaches were also considered as an important string matching technique for analyzing biological sequences. The process in this technique simulates non- deterministic finite automata in performing parallel bit operations in a computer words per clock cycle (Baker, 1993). These bits operations are scaled down to minimal level thus, improving the performances of the string pattern matching process (Ďurian et al., 2009; Hazay, 2004). This technique also helps to improve the efficiencies of the early character based pattern matching algorithms like Knuth-Morris-Pratt algorithm (KMP), Boyer-Moore (BM), BMH, BMHS etc. because of their incapacity to be efficiently applied on biological sequence and their inability to cope with larger sequences.

Numerous bit-parallel string pattern matching algorithms were proposed for single or multiple patterns and also for exact and inexact (approximate) pattern matching. In exact pattern matching, we look for the exact occurrences of a given pattern in large body of text while in an approximate pattern matching, we look for the occurrences of a pattern with some modification in a large boy of text. For example, after the first proposed method called Shift-OR algorithm, an approximate multi-pattern algorithm was introduced in (Salmela et al., 2007). An exact single pattern matching algorithm using Backward non-deterministic machine (BNDM) where AND and/or Shift operations are carried out concurrently was introduced by (Mendivelso et al., 2020). An advanced method named two- way non- deterministic machine (TNDM) was introduced by Anderson-Lee et al (Anderson-Lee et al., 2016), unlike in BNDM forward operation is always carried out instead of shifting operation whenever a mismatch occur at last position (Gupta & Rasool, 2014; Islam & Talukder, 2017). An enhanced exact single pattern matching algorithm that combine wide window and bit parallelism approaches was introduced in He, Fang, and Sui (He et al., 2005). Hybrids approach that reduces the number of false matches using Shift-OR algorithm with Q-gram proposes an approximate multi-patterns matching algorithm that matches one character at a time in (Mauri & Pavesi, 2005). Similarly, Peltola and Tarhio (2003) proposes using Q-gram and BNDM to implement exact multi-pattern string matching algorithms BNDMq and SBNDMq that read one character at a time at the beginning of each alignment before testing the state variable (Peltola & Tarhio, 2003). An efficient bit-parallel algorithm for alpha and delta matching that combine both dynamic programming and bit-parallelism approaches for music retrieval was proposed in (Heyne et al., 2009). Although these algorithms have achieved significant improvements on performance, the objective is to have more efficient method that can optimize the performances of the previous methods and be conveniently used to analyze biological data. Therefore, this paper shows how bit-parallelism approach can be used achieve this objective. Therefore, this paper seek to provide a comprehensive comparative analysis of bit-parallel string pattern matching algorithms in order to assist the researchers in this area to easily select the most appropriate method for a particular application area (Boni & Gunn, 2021).

2. Bit-Parallel Pattern Matching Algorithms for Biological Sequence

Bit-parallel string pattern matching algorithms are proposed for exact or inexact (approximate). In exact matching, the process finds all exact occurrences of a given n-length pattern in an m-length text while in an approximate (inexact) pattern matching, sometime referred to as matching with certain number of mismatches, the process is to locate the occurrences of substring X in an m-length string T that are similar to a given n-length pattern P with a limited number of say K different characters in similar matches.

Bit-parallel pattern matching can involve single or multiple patterns. In a single pattern matching only one pattern is involved while in multi-patterns matching, the process can be in either of these scenarios. Classes of characters, here the text matches the character at position j in any of the patterns. For example, let **P1**=CAGT and **P2**=AGTC be two distinct patterns of equal length then we can form the supper pattern **P'**={CA}, {AG}, (He et al.), {TC} from **P1** an **P2** that matches the text strings "CAGT", "CGTC", "AGTC" and "AGCT". This approach is more suitable for matching with large

number of patterns or concatenating the 1st, 2nd, 3rd etc. characters of each pattern to form a single pattern. For example, let P1=ACTGand P2=GACT be two distinct patterns of equal length then we can form a single patternP'= AGCATCGT and concatenating the patterns. For example, Let P1=ACTG and P2=GACT be two distinct patterns of equal length then we can form a single pattern P'=ACTGGACT.

In bit-parallel string-matching algorithm, the processes are done in pre-processing and matching phases. At the pre-processing phase, bit mask **B[c]=b_m ...b_1**. For each symbol **c** in the text is computed as the ith bit of each character is set to **0** if the character appears at position **i** else is set to **1** written from right-to-left and table **B** is then constructed to store these bit masks. During the search process, the state of the automaton is kept as state vector in a machine word **D = d_m...d_1** and transit from state **i** state i+1 when the ith bit of the bit mask **B[C]** of the character **C** is zero. The automaton is then set active at state **i** if and only if the ith bit of the state vector D matches the end of the string read up to the position **i.** The state vector is initially set **D=0_m** and for each new character t_j input, the state vector is updated using the formula **D← ((D<<1) |0^{m-1} 1) &B[t_j],** for **m** ≤ **w** where **m** is the pattern length and **w** is the computer word length. A match occurs at position **i** whenever the main significant bit of **D** is **0**.

2.1 Shift-OR algorithms

Shift-And algorithm first build a table for each character of text in a bit mask **b_m...b_1,** the mask has the ith set if and if **P=C**. the state of the search is kept in a machine word **D = d_m...d_1**, **d** is set when **p** matches the text end up to current position. Match is reported whenever **d_m** is set. Initially the machine word is set to **D=0^m** and then for each new character t_j, **D** is updated using the formula **D= ((D≪ 1) | 0^{m-1} 1) & B[t_j]**. This algorithm works only when **m** ≤ **n** where **m** and **n** are text length and pattern length. Bit parallel shift-And algorithm is a simpler variant of shift-Or with some further advantage of being easily extended to classes of characters i.e. it can be extended to handle wild card character, regular expression approximate search.

In Shift-Or bit mask of table **B** are complemented and ≪ operator resulted in an empty string ∅ to the right of **D'** and the suffixes stemming from the empty string is already in **D'**. The operator **|** is used instead of **&** thus the recursive becomes **D'= (D≪ 1) | B[t]** and a match is reported when **d_m=∅.**

2.2 Backward Non-deterministic Matching (BNDM)

Algorithms based on this approach simulate the suffix automaton of x^R. The process construct a table for each character **c** to stores the bit mask. The bit-mask in B_c is set iff x_i **= c** and the search state is kept in a word d = d_{m-1}... d_0. The bit d_i at iteration **k** is set when x[m - i … m-1-i+k] = y[j + m-k .. j+m-1]. At iteration **0**, **d** is set to 1^{m-1} and the vector is updated as **d'** = (d & B[yj]) << 1. Matched is reported only when d_{m-1}**=1** after iteration **m** hence, pattern prefix matched in the current window position **j** and the longest prefix matched gives the shift to the next position. This algorithm is most efficient when the pattern length is not longer than the computer word size.

2.3 Two-way modification of BNDM (TNDM)

This is the variant of BNDM algorithm that searches the text string so that, given

an m-length text T string and n-length string pattern P, comparison between T and P is achieved if the sub-string ti $t_i \in$ T = P_n or Sub-string $t_i \in$ T exist in P_n and $t_i \neq P_n$ and Sub- string $t_i \in$ T does not exist in P_n. The algorithm works as in BNDM when the first and third conditions hold while TNDM searches in the forward direction to get the next occurrence when the second condition holds thus, decreasing the number of searched characters. Therefore, TNDM algorithm searches the text string character by character in the forward direction until it get to a character k such that:

I. Sub-string $\mathbf{t_i ... t_k}$ does not exist in $\mathbf{P_n}$ or
II. Sub-string $\mathbf{t_i ... t_k}$ forms a suffix of $\mathbf{P_n}$

In case I, the shifting can go beyond the previous alignment of P while, in case II, the algorithm continues to search backward from the text position i-1 i.e. reverting back to BNDM algorithm.

2.4 Simplified BNDM algorithm (SBNDM)

This is another BNDM variant, the algorithm works as in BNDM during the shifting process. Match is reported when the sub-string th…ti does not appear in P i.e. if h < j $\leq$ i and t j…ti are the prefixes of P then the next alignment starts at j where j is the smallest index in P then the next alignment starts at i+1 and the scanning of prefixes is skipped so that the starting position of the next alignment is now set to h. This simplifies the running of the inner loop of the algorithm and resulted in a reduced average shifting length.

3. New trend in Bit-parallel string pattern matching

The use of Bit-parallel string pattern matching algorithms has been a greater tool for solving problems of bio-sequence analysis. However, biological sequences such as that of RNA and protein have the ability to fold and form some complex structures that are critical to their functions. The complexities of these structures have made the benchmark bit-parallel algorithms unable to efficiently make their functional inferences (Prasad & Agarwal, 2008). This is because; their nucleotide strings are complementary to one another. To deal with this, the parameterized string matching concept was first introduced in (Das & Kapoor, 2017; Kumar et al., 2010). In this concept, given two strings X and Y then we can say that the two strings are equivalent though not identical if Y can be obtain by simply renaming some of the variables of X and vice vasa (Prasad, 2016). This type matching will be unsuccessful with benchmark algorithms because the two strings are not identical.

3.1 Parameterized Bit-Parallel pattern matching for biological sequences

This is an efficient method that was first used in software maintenance to find an equivalency between two program codes and was later used as a tool to deal with some problems in biological sequence analysis. In parameterized matching, the process grouped the strings into set of fixed or constant alphabets $\sum$c (whose symbols cannot be modified) and that of parameter alphabets $\prod$ (whose symbols can be modified) i.e. two strings **X** and **Y** are said to be parameterized matched if and only if some symbols in **Y** can be transformed or modified to make **Y** equal to **X**. In this type of matching, the occurrences of a pattern **P** in a text string **T** is reported only when the

constant symbols exactly matched and the parameter symbols are renamed or modified.

For example**, let Σ = {A, B},* ∏*={X, Y, Z, W} and P=*XAYBX *with parameter matching where f(X ↔Z; Y↔W) then P matches the text* ***T*= ZAWBZ.*

Therefore, we can define parameterize matching as:

Let ∑ *and* ∏ *be two disjoint finite set of fixed and parameter alphabets respectively, then two p-strings X and Y over an* ***m****-length alphabet (*∑ ∪ ∏*) are said to be p-*matched *if there exist a bijective function say f:* ∑ ∪∏ ↔ ∑ ∪ ∏ *such that X[i] = f(Y[i]) where 1* ≤ *i* ≤ ***m*** *and in particular* ***f*** *established an identity mapping for the symbols* ∏. For parameterized matching for biological sequences, the set of constant alphabet is empty (i.e. ∑*= Ø*). The efficiency of the String pattern matching algorithms is greatly affected by the string encoding used. For instance variable width encoding when used can slow down the matching process. In this case the speed can be enhanced by searching the sequence unit instead, though it may result in having false matches if not specifically designed to avoid it (Prasad, 2016). The breakthrough on p-matching by Baker was achieved with introduction of an encoding scheme called the previous encoding (prev- encoding for short) as formally defined:

Prev-encoding: *Given an n-length p-string T and non-negative integer F, the mapping (*∑∪ ∏*)*→(*∑∪ *F)* encodes each constant alphabets say C* ∈∑ *with the* same *symbol C and each parameter symbol say z* ∈ ∏ *to the distance from its previous z, here* ∑ *=* ***Ø*** *in case of biological sequences (Prasad et al., 2011).*

There are variants of Parameterized Bit-parallel string matching algorithms that proposed to handle exact or approximate bit-parallel matching with either single or multiple patterns such as Parameterized shift-Or and shift-And as in Kumar et al (Kumar et al., 2010) and parameterized shift-Or and shift-And extended with super alphabet as in Kesavaraj and Sukumaran (Kesavaraj & Sukumaran, 2013). Exact parameterized matching Beal and Adjeroh (Beal & Adjeroh, 2015), approximate Hakak et al (Hakak et al., 2019); Hakak et al (Hakak et al., 2019), compressed Xu et al (Xu et al., 2005) and multi-pattern matching Gharib et al were also proposed (Gharib et al., 2008).

3.2 Parameterized bit-parallel Shift-OR algorithm

The process is similar to that of standard shift-Or algorithm but in this case, the previous encoding for the pattern and text are first computed at the pre-processing stage and these encodings are considered for matching in at the searching phase. During the matching, match is reported only when the previous encoding of the pattern matched the previous encoding of the text suffixes i.e.

Let ***P*** *be a p-string pattern and* ***T*** *a p-string text then* ***P*** *p-matches sub-sting of T at position* ***i*** *if and only if* ***prev(P)*** *is a prefix of* ***p-suffix(T, i)****.*

For parameterized Shift-And set D=0^m originally and, for each new character t_j, update D using the formula D'← ((D<<1)|0^{m-1}1) & B [t_j]

3.3 Parameterized Bit-parallel BNDM Algorithm

The process is the same as that of the standard BNDM algorithm but the use previous encoding is introduced as in Prasad et al (Prasad et al., 2011). The encodings

for both the strings pattern and text are first computed at the pre-processing stage. Other variant of BNDM algorithms that can be used to handle both parameterized, exact or approximate bit parallel matching with either single or multiple patterns include: Parameterized BNDM, TNDM (Two way modification of BNDM) and Simplified BNDM (SBDM) as in Anderson-Lee et al (Anderson-Lee et al., 2016) and BNDM with q-gram as in Kumar et al (Kumar et al., 2010) etc.

4. Performance Analysis of Bit-parallel String matching Algorithms

Many bit-parallel String pattern matching algorithms for similarity comparison on biological sequences have been proposed each with its own area of application, strength and weaknesses as shown in **Table 1**. Due to the rapid generation of the biological sequences, choosing and understanding an appropriate string matching algorithm to use in addressing challenges in a particular application area becomes difficult as the process involves large amount of computations that consume a lot of time (Zhong et al., 2010). Thus, choosing an appropriate algorithm to use should be based on its applicability and complexity in the application area in question. Though, the performances achieved with benchmark bi-parallel algorithms were appreciable, the implementation of some of these algorithms on biological sequences such as RNA without modification was theoretically bad. In general, optimal performances can only be achieved with these algorithms when their development focuses on the nature of the target biological sequence.

Table 1: Performances of Bit-parallel String matching Algorithms

Algorithms	Performance	Drawback	Application Areas
(Peltola & Tarhio, 2003)	O(n[m/w](w + m - 1)) worst case and O(δ x (w +m - 1)) extra space	Limited to few data. (Külekci, 2008)	Natural language, DNA sequence, and binary alphabet
(Peltola & Tarhio, 2003)	O(n[m x w]) worst case with (δ[m/w]) extra space.	Works well when searching for a complete match.	DNA and natural English text
(Grabowski & Fredriksson, 2008)	Average and worst case:O(n). Total average time: $O(n\log_{\delta}(m/m))$	Not good for shorter patterns (Faro et al., 2012)	DNA sequence, natural text, binary input
(Alqadi et al., 2007)	O(m / n)	Not good for non-uniform patterns.	DNA sequence
(Grabowski & Fredriksson, 2008)	O (n⌈ m/w⌉) worst case	Not good for long sequence	DNA and Text
(Prasad & Agarwal, 2008)	Linear	For Parameterized strings	Text and DNA
(Zhang et al., 2009)	Pre-processing: O(mδ). Searching: O(nm)	Performance degrades with changes in patterns length.	DNA sequence
(Ďurian et al., 2010)	Not available	Good for long pattern	All Biosequences
(Prasad et al., 2011)	Not available	Only for approximate matching	All Biosequences
(Faro et al., 2012)	O(nm) and O(δ) for extraspace when $O(m \leq w)$	Performance depends on pattern length	English text, DNA sequence and binary data
(Prasad, 2016)	Linear	Good for parameterize matching	All Biosequences

5. Conclusion

Bit-Parallelism plays a vital role in string comparison and pattern detection which makes it a versatile tool that can be applied in any form of biological sequence analysis such as, Characterization and/or annotation a newly discovered biological sequences, finding sequence or structural homology between biological sequences etc. Though, it has good application in biological sequence analysis, its application was mostly hampered by the limitation on the pattern length that should be less than or equal to the computer word length (Beal & Adjeroh, 2016). How to use the bit-parallel approach to find structural similarities between Biological sequences such that RNA whose characteristics differ from that of other biological sequences such as DNA could be the task for future research.

References

Allali, J., & Sagot, M.-F. (2005). A new distance for high level RNA secondary structure comparison. *IEEE/ACM transactions on computational biology and bioinformatics*, *2*(1), 3-14. https://doi.org/10.1109/TCBB.2005.2

Alqadi, Z. A., Aqel, M., & El Emary, I. M. (2007). Multiple Skip Multiple Pattern Matching Algorithm (MSMPMA). *IAENG International Journal of Computer Science*, *34*(2), 03. https://www.iaeng.org/IJCS/issues_v34/issue_2/IJCS_34_2_03.pdf

Amir, A., & Nor, I. (2007). Generalized function matching. *Journal of Discrete Algorithms*, *5*(3), 514-523. https://doi.org/10.1016/j.jda.2006.10.001

Anderson-Lee, J., Fisker, E., Kosaraju, V., Wu, M., Kong, J., Lee, J., Lee, M., Zada, M., Treuille, A., & Das, R. (2016). Principles for predicting RNA secondary structure design difficulty. *Journal of Molecular Biology*, *428*(5), 748-757. https://doi.org/10.1016/j.jmb.2015.11.013

Baker, B. S. (1993). A theory of parameterized pattern matching: algorithms and applications. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing* (pp. 71-80). http://softalytics.com/papers/STOC93.ps.gz

Baker, B. S. (1995). Parameterized Pattern Matching by Boyer-Moore-Type Algorithms. In *In Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms* (Vol. 95, pp. 541-550). https://dl.acm.org/doi/pdf/10.5555/313651.313816

Beal, R., & Adjeroh, D. (2015). Efficient pattern matching for RNA secondary structures. *Theoretical Computer Science*, *592*, 59-71. https://doi.org/10.1016/j.tcs.2015.05.016

Beal, R., & Adjeroh, D. (2016). Compressed parameterized pattern matching. *Theoretical Computer Science*, *609*, 129-142. https://doi.org/10.1016/j.tcs.2015.09.015

Boni, A., & Gunn, M. (2021). Building and leveraging the innovation ecosystem and clusters: universities, startups, accelerators, alliances, and partnerships: A "From the Boardroom" Perspective by the Special Edition Co-Editors. *Journal of Commercial Biotechnology*, *26*(1). https://doi.org/10.5912/jcb963

Cantone, D., & Faro, S. (2014). Efficient Online Abelian Pattern Matching in Strings by Simulating Reactive Multi-Automata. Stringology,

Das, S., & Kapoor, K. (2017). Weighted approximate parameterized string matching. *AKCE International Journal of Graphs and Combinatorics*, *14*(1), 1-12. https://doi.org/10.1016/j.akcej.2016.11.010

Ďurian, B., Holub, J., Peltola, H., & Tarhio, J. (2009). Tuning BNDM with q-grams. In

2009 Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments (ALENEX) (pp. 29-37). SIAM. https://doi.org/10.1137/1.9781611972894.3

Ďurian, B., Peltola, H., Salmela, L., & Tarhio, J. (2010). Bit-parallel search algorithms for long patterns. In *International Symposium on Experimental Algorithms* (pp. 129-140). Springer. https://doi.org/10.1007/978-3-642-13193-6_12

Faro, S., Lecroq, T., Holub, J., Watson, B., & Žďárek, J. (2012). Twenty years of bit-parallelism in string matching. *Festschrift for Borivoj Melichar*, 72-101. https://igm.univ-mlv.fr/~lecroq/articles/BM70.pdf

Gharib, T. F., Salah, A., El-Henawy, I. M., & Salem, A.-B. M. (2008). Protein Structure Searching using Suffix Arrays. In *In Proceedings of the 2008 International Conference on Bioinformatics and Computational Biology, BIOCOMP 2008* (pp. 688-691). https://www.researchgate.net/publication/221051767

Grabowski, S., & Fredriksson, K. (2008). Bit-parallel string matching under Hamming distance in O (n⌈ m/w⌉) worst case time. *Information Processing Letters*, *105*(5), 182-187. https://doi.org/10.1016/j.ipl.2007.08.021

Gupta, S., & Rasool, A. (2014). Bit Parallel String Matching Algorithms: A Survey. *International Journal of Computer Applications*, *95*(10), 27-32. https://doi.org/10.5120/16632-6501

Hakak, S. I., Kamsin, A., Shivakumara, P., Gilkar, G. A., Khan, W. Z., & Imran, M. (2019). Exact string matching algorithms: survey, issues, and future research directions. *IEEE Access*, *7*, 69614-69637. https://doi.org/10.1109/ACCESS.2019.2914071

Hazay, C. (2004). Parameterized Matching. In String Processing and Information Retrieval, 15th International Symposium, SPIRE 2008, Melbourne, Australia. *Bar-Ilan University*, 236-248. https://doi.org/10.1007/978-3-540-89097-3_23

He, L., Fang, B., & Sui, J. (2005). The wide window string matching algorithm. *Theoretical Computer Science*, *332*(1-3), 391-404. https://doi.org/10.1016/j.tcs.2004.12.002

Heyne, S., Will, S., Beckstette, M., & Backofen, R. (2009). Lightweight comparison of RNAs based on exact sequence–structure matches. *Bioinformatics*, *25*(16), 2095-2102. https://doi.org/10.1093/bioinformatics/btp065

Islam, T., & Talukder, K. H. (2017). An improved algorithm for string matching using index based shifting approach. In *In 2017 20th International Conference of Computer and Information Technology (ICCIT)* (pp. 1-4). IEEE. https://doi.org/10.1109/ICCITECHN.2017.8281772

Kesavaraj, G., & Sukumaran, S. (2013). A study on classification techniques in data mining. In *In 2013 fourth international conference on computing, communications and networking technologies (ICCCNT)* (pp. 1-7). IEEE. https://doi.org/10.1109/ICCCNT.2013.6726842

Külekci, M. O. (2008). A method to overcome computer word size limitation in bit-parallel pattern matching. In *Algorithms and Computation: 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings 19* (pp. 496-506). Springer. https://doi.org/10.1007/978-3-540-92182-0_45

Kumar, K., Prasad, R., & Agarwal, S. (2010). Software maintenance by multi-patterns parameterized string matching with q-gram. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 1-5. https://doi.org/10.1145/1764810.1764822

Mauri, G., & Pavesi, G. (2005). Algorithms for pattern matching and discovery in RNA secondary structure. *Theoretical Computer Science*, *335*(1), 29-51. https://doi.org/10.1016/j.tcs.2004.12.015

Mendivelso, J., Thankachan, S. V., & Pinzón, Y. (2020). A brief history of parameterized

matching problems. *Discrete Applied Mathematics*, *274*, 103-115. https://doi.org/10.1016/j.dam.2018.07.017

Miao, C., Chang, G., & Wang, X. (2010). Filtering based multiple string matching algorithm combining q-grams and bndm. In *In Fourth International Conference on Genetic and Evolutionary Computing* (pp. 582-585). IEEE. https://doi.org/10.1109/ICGEC.2010.149

Peltola, H., & Tarhio, J. (2003). Alternative algorithms for bit-parallel string matching. In *In String Processing and Information Retrieval: 10th International Symposium, SPIRE* (pp. 80-93). Springer. https://doi.org/10.1007/978-3-540-39984-1_7

Prasad, R. (2016). Efficient Parameterized Word Matching Using Bit-Parallelism and Partitioning the Text. *International Research Journal of Electronics and Computer Engineering*, *2*(2), 20-24. https://core.ac.uk/download/pdf/230309917.pdf

Prasad, R., & Agarwal, S. (2008). Parameterized shift-and string matching algorithm using super alphabet. In *In 2008 International Conference on Computer and Communication Engineering* (pp. 937-942). IEEE. https://doi.org/10.1109/ICCCE.2008.4580744

Prasad, R., Sharma, A. K., Singh, A., Agarwal, S., & Misra, S. (2011). Efficient bit-parallel multi-patterns approximate string matching algorithms. *Scientific Research and Essays*, *6*(4), 876-881. https://academicjournals.org/journal/SRE/article-full-text-pdf/3C2654423657

Salmela, L., Tarhio, J., & Kytöjoki, J. (2007). Multipattern string matching with q-grams. *Journal of Experimental Algorithmics (JEA)*, *11*, 1.1-es. https://doi.org/10.1145/1187436.1187438

Xu, Y., Wang, L., Zhao, H., & Li, J. (2005). Exact matching of RNA secondary structure patterns. *Theoretical Computer Science*, *335*(1), 53-66. https://doi.org/10.1016/j.tcs.2004.12.016

Zhang, G., Zhu, E., Mao, L., & Yin, M. (2009). A bit-parallel exact string matching algorithm for small alphabet. In *Frontiers in Algorithmics: Third International Workshop, FAW 2009, Hefei, China, June 20-23, 2009. Proceedings 3* (pp. 336-345). Springer. https://doi.org/10.1007/978-3-642-02270-8

Zhong, N., Li, Y., & Wu, S.-T. (2010). Effective pattern discovery for text mining. *IEEE transactions on knowledge and data engineering*, *24*(1), 30-44. https://doi.org/10.1109/TKDE.2010.211