ISSN 2347 - 3983
Volume 8. No. 3, March 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter13832020.pdf
https://doi.org/10.30534/ijeter/2020/13832020

# Local Concurrency in Text Block Search Tasks

**Olesia Barkovska[1], Oleg Mikhal[2], Daria Pyvovarova[3], Oleksii Liashenko[4], Vladyslav Diachenko[5], Maxim Volk[6]**
[1,2,3,4,5,6] Department of Electronic Computers
Kharkiv National University of Radio Electronics, NURE
Kharkiv, Ukraine
d_ce@nure.ua

**ABSTRACT**

The relevant problem of text analytics tools development is real-time mode establishment via data processing operations parallelizing. The use of the local parallel (LP) approach is advantageous. The classification of search algorithms for text blocks, which are candidate for local parallel (LP) approach implementation, was presented. The advantages of LP approach implementation were regarded. The variants of bit and segmental word-patterns processing in text block search tasks were proposed and exemplified. Perspectives and certain threats to its implementation were studied in terms of Boyer-Moore algorithm.

**Key words:** Boyer-Moore algorithm, local parallel data processing, text analytics, text block search.

## 1. INTRODUCTION

Block search (of a word or phrase) in a text is a common task of the algorithm theory, which combines intuitive statement understandability and overall concept simplicity with multivariance of solutions depending on the auxiliary conditions [1, 2] with regard to particular features [3 - 5] of the information to be processed.

Text block search algorithms have been studied in sufficient detail [6], in particular, for sequential computer systems [7]. Parallel computer systems were until recently predominantly multi-processor, cumbersome, expensive and “special-purpose”. Therefore, they were developed mainly to solve particular tasks and the problem of sorting and search was regarded primarily in terms of common algorithm parallelizing principles realization in respect to data bulk.

The situation changed “at the cusp of centuries” with the emergence of multicore processors and mass use of network computation structures [4, 5]: limited (relatively small) data arrays processing with general-purpose computers prior use has gained relevancy. It is quite clear that limited data arrays and general-purpose computers of the early XXIst century are almost like big data and back-end processors of the late XXth century. Figure 1 shows basic sequential (elementwise) text block search algorithms classification, which cannot be considered complete. The given task class is effectively solved by means of using algorithms, which realize the principle of local parallel (LP) data processing [4]. On the basis of each of the abovementioned algorithms, a corresponding LP version may be developed.

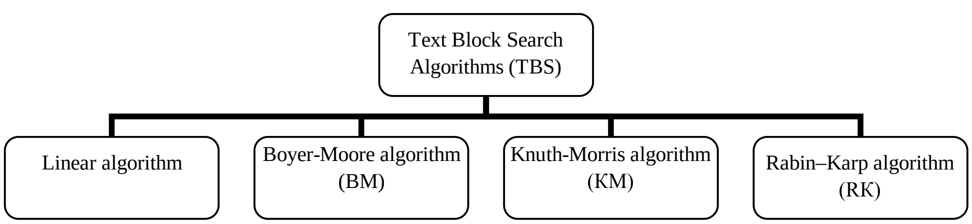


**Figure 1:** Review of basic TBS algorithms

## 2. RESEARCH TASK RATIONALE

Limited data arrays (LDA) refer to what an individual operates with while interacting with the environment. Preferential LDA application field covers the decision-making tasks in the determinated set of options. The tasks of the given type cover, in particular, the major part of the subject of man-machine interaction. One of the dominant paradigms (not the only one [5]) in this field is the research and reproduction of the structure and functions of the human intellect [9]. At every certain moment of their practical activity, an individual operates effectively with a limited small number of notions [10, 11]. Excess LDA is a stress situation, in which effective work may demand specific training [3, 10] or may be supported, at the best, during a limited time period. Effectiveness is determined by the correct choice and the effective sequence of options consideration while the effective choice effectuation eventually reduces itself to the tasks of TBS type. In the given tasks, LP algorithms provide for acceleration of processing with no extra hardware resources application. Thus, LP processing is feasible as a structural element in TBS implementation. This stipulates the relevancy and practical value of the given direction.

## 3. AIMS AND TASKS OF THE WORK

The aim of the given work is the study of LP data processing principles as applied to TBS tasks. The specificity of the known algorithms (Fig. 1) is the sequential (elementwise) study of the material in implementation on general-purpose computer systems. The LP approach enables to parallelize the study within the framework of LDA. With regard to this, the text block search principles are studied as implemented in LP algorithms:

– traditional TBS algorithms classification is provided (with account taken of peculiarities important for further LP realization);

– the possibilities for LP approach application are analyzed (with regard to the developments [4]);

– the options of bit and segmental LP lines (word patterns) processing are outlined;

– the implementation options for LP algorithms of fragmental search (by word patterns) are regarded, in particular, based on Boyer-Moore algorithm.

## 4. TASK FULFILLMENT

Text analytics tools enable automatic choice, systematization and analysis of alphanumeric data due to the application of linguistic rules, statistical machine learning methods within various areas of application [3, 9, 10, 11]. TBS is a particular case of text analytics tasks, i.e. the search of information (word patterns) in alphanumeric data storages (in texts). [1] offers the method of parallelizing the algorithm of word pattern search in the text for shared memory systems, which is based on adaptive data input decomposition. However, despite the obtained reduction of time for the set task accomplishment, the real-time mode cannot yet be reached.

Reduction of time for word pattern search in the text is possible with the account taken of the allowed relations of between the processor capacity, word size of the LP-representation and the possible maximum permissive amount of alphabetic characters.

Local parallel (LP) arrangement of computations can be outlined as follows [4]: assuming there are two n-component vectors: A: $\{a_1, a_2,\ldots, a_i,\ldots, a_n\}$; B: $\{b_1, b_2,\ldots, b_i,\ldots, b_n\}$; $i \in (1, 2,\ldots, n)$; $0 \le a_i \le a_{max}$; $0 \le b_i \le b_{max}$; $a_{max}$, $b_{max}$ are integer; $a_{max} = b_{max}$. Here, the prompts $a_{max}$ и $b_{max}$ define that the numbers' magnitude (vector components) is limited (are not arbitrary large). Component-based vector sum:

C: $\{ c_1, c_2,\ldots, c_i,\ldots, c_n \}$; $c_i = a_i + b_i$, is required to be found by means of applying a computing medium with a general purpose processor. Thus, this is not a question of special-purpose multiprocessor systems.

In the traditional sequential variant, the number of operations is proportional to n:

1. Initial installation: i=1;
2. Operand $a_i$ value retrieval;
3. Operand $b_i$ value retrieval
4. Summarization: $c_i = a_i + b_i$;
5. Storage of the result $c_i$ in the holding register;
6. i = i + 1. with i>n, termination; otherwise, - point 2

In LP variant, the computation scheme is different:

1. Concatenation:

$$A:\{a_1, a_2,\ldots, a_n\} \rightarrow a_\# = (a_1 \oplus a_2 \oplus \ldots \oplus a_n);$$

$$B:\{b_1, b_2, \ldots, b_n\} \rightarrow b_\# = (b_1 \oplus b_2 \oplus \ldots \oplus b_n).$$

2. Concatenant $a_\#$ retrieval;
3. Concatenant $b_\#$ retrieval;
4. Concatenants summarization: $c_\# = a_\# + b_\#$;
5. Storage of the result $c_\#$ in the holding register
6. Deconcatenation: $c_\# = (c_1 \oplus c_2 \oplus \ldots \oplus c_n) \rightarrow C:\{c_1, c_2,\ldots, c_n\}$.

Concatination and deconcatination operations are performed with positive integers in binary representation. These integers reside in the neighbouring non-overlapping segments. Therefore, figures $a_\#$, $b_\#$ and $c_\#$ are register forms (RF). In processing, they reside in the central processing unit in separate registers – lines of binary memory cells. In LP representation, the concatenation result is a positive integer interpreted as a composition of segments with regard to concatenation length. Therefore, concatenation is the operation of packing information representation in a# and b#. in the form of segments. Further, are processed as a whole as numerals. In deconcatenation, segments are extracted in separate variables. Computing blocks of pp 2-5 in the sequential and LP schemes coincide, however, in the sequential scheme, the block is executed n-fold and in the LP scheme it is executed one-shot. This stipulates the advantage in effectiveness. Concatenation and deconcatenation are connected with extra computing power consumption. On the other hand, if something more complex (multi-stage) than in p.2-5 is effectuated in the system with the results retrieved at the end, extra time spending may be negligible as compared to the overall computational cost. Therefore, the effectiveness of the LP scheme increases pro rata to the number of concatenants n.

The alphabet size (the number of characters) is another important factor. The ultimate possible alphabet size for segments packing in LP word coding in register representation (RP) is defined by the ratio: N = 2n,in which : N is the number of alphabetic characters; n is the word size in bits. For several values of the M processor register capacity, Table 1 presents the correlation of ultimate segment packing

RP and the processor word size. The number of bits necessary to represent numbers, which corresponds to the number of the position of ultimate word pattern shift in the BM algorithm [7] is given after the slash.

Some of the values in Table 1 can be smaller than "the available" ones with account taken of the fact that implementation of LP-algorithms requires the use of additional uppermost bits as "technological bits". The table distinguishes acceptable (feasible for use) options. As a whole, the table presents ultimate n word lengths (in bit representation), which can be input in the register of M processor capacity.

**Table 1:** Correlation of RP segment number with n word size and M processor capacity

| **M** | **n** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **16** | 5 / 3 | 3 / 2 | 3 / 2 | 2 / 1 | 2 / 1 | 1 / 1 | 1 / 1 |
| **32** | 10 / 4 | 7 / 3 | 6 / 3 | 5 / 3 | 4 / 2 | 3 / 2 | 3 / 2 |
| **64** | 21 / 5 | 15 / 4 | 12 / 4 | 10 / 4 | 9 / 4 | 7 / 3 | 7 / 3 |
| **128** | 42 / 6 | 31 / 5 | 25 / 5 | 21 / 5 | 18 / 5 | 15 / 4 | 14 / 4 |
| **256** | 85 / 7 | 63 / 6 | 51 / 6 | 42 / 6 | 36 / 6 | 31 / 5 | 28 / 5 |

In order to analyze and demonstrate the principles of LP-variant construction for a BM algorithm, the option of a 32-bit processor with 4-bit word size the was selected. The given choice is explained by compact and well-observed expressions in the demonstration of algorithms. As shown in Table 1, this corresponds to the 16-character alphabet. Practical application of the given option is evidently limited to a certain extent. However, the purpose of the proposed illustration is the demonstration of the operation principle and not the direct practical use.

The LP variant has minimum two extra opportunities to improve the algorithm of character search in a line related to bit-by-bit and segmental comparison. Bit-by-bit comparison enables to reveal the irrelevance of bit lines representation and segmental comparison helps to find irrelevance of separate segments i.e. symbols in LP representation. The corresponding algorithms for MB bit-by-bit comparison pattern and MS segmental comparison pattern are to be studied further.

The compared pattern and the line fragment must be primarily converted into LP representations. This demands the symbol codes to be stored in neighbouring non-overlapping segments. The sequence of segments stored in each RP must further be interpreted as numbers in bit representation.

The situation can be exemplified. Given the presence of a 7-character alphabet

$$(a, b, c, d, e, f, g). \qquad (1)$$

We are confined to the 7-character alphabet for the purpose of conciseness and visual clarity because only 3 bits are required to code these symbols:

| **a** | **b** | **c** | **d** | **e** | **f** | **g** |
|---|---|---|---|---|---|---|
| 001 | 010 | 011 | 100 | 101 | 110 | 111 |

(2)

Code 000 is not used as the alphabet character because it is used to represent disalignment of the segment meaning. In RP, the 7-character alphabet may contain 10 3-bit segments for a 32-bit processor.

Let us assume that there is a text block, pattern = (bcdabcbcbccfeggeadda) written in 7-character alphabet (1). There are no blank spaces because the space character (1) is not planned. Let us assume that there is a word pattern str = (bcbcbccfeg) to be searched in the text. It can easily be seen that str resides in the line pattern with the shift of 5, but we are not yet interested in search but the demonstration of bit-to-to and segmental comparisons.

With the account taken of (2), the word pattern *str* is coded in RP:

$$(010)(011)(010)(011)(010)(011)\ (011)(110)(101)(111) \qquad (3)$$

Parentheses punctuator in (3) is only used in order to demonstrate segmentation in RP. In the processor register this is a prime number. The subscript hereinafter represents the numerical system processor word.

$$(010011010011010011011110101111)_2 = (323827631)_{10}. \qquad (4)$$

Bit-to-bit comparison of RP *str* (4) and RP of the segment (the first 10 characters) in the line *pattern* with zero shift is required

$$(010011100001010011010011010011)_2 = (327496915)_{10} \qquad (5)$$

For this purpose, two invariables are defined:

B1 = $(010101010101010101010101010101)_2$ = $(357913941)_{10}$;

B2 = $(101010101010101010101010101010)_2$ = $(715827882)_{10}$,

which can enable fulfillment of a series of logical operations. In B1 ones take odd-numbered positions and in B2 they take even-numbered positions.

Verification:

B1 + B2 = $(357913941)_{10} + (715827882)_{10}$ =

= $(1073741823)_{10} = (111111111111111111111111111111)_2$

With the help of B1 and B2, str (4) and pattern (5) are decatinated into odd-numbered (str1 and pattern1) and even-numbered (str2 and pattern2) binary positions:

$000100010100010100010101000000101)_2 = (289740037)_{10}$ ;

pattern1 =pattern AND B1 = $(000100010000010100010100010100 01)_2$ =$(285545553)_{10}$;

str2 = str AND B2 = $(001000000100000100010101010 10)_2$ = $(34087594)_{10}$ ;

pattern2 = pattern AND B2 = $(001010000000000100000010000010)_2$ = $(41951362)_{10}$.

Verification:
str1 + str2 = $(289740037)_{10}$ + $(34087594)_{10}$ = (323827631)10 = str, (see (4))
pattern1 + pattern2 = (327496915)10 = pattern, (see (5))
Arithmetic summarization is conducted for (str1+ pattern1) and (str2 +pattern2). Bit-by-bit pairs (1, 0), (0, 1) и (0, 0) remain unchanged, and in pairs (1, 1) one shifts to a senior bit. These shifted ones are reduced by B2 for (str1+ pattern1) and B1 for (str2 +pattern2). Then we obtain bit-by-bit comparison pattern:
MB = (B2 AND (str1+ pattern1)) + (B1 AND (str2 +pattern2)) =
= $(000000110010000000001101111100)_2$.
In this pattern, ones represent non-overlapping bits (bit-by-bit combinations (1, 0) и (0, 1)).
Bit-by-bit comparison is not of interest in the case under study alone, but as a component of segmental comparison. Segments (i.e. character codes in RP (4) и (5)) do not overlap if there are ones in the corresponding positions in MB patterns.
In order to build MS segmental comparison pattern, we input 4 invariables: F1 and F2 in order to separate odd and even segments; L1 and L2 in order to set additional (“technological”) ones. In F1, the ones reside in odd segments and in F2 they reside in even segments:
F1 = $(000111000111000111000111000111)_2$ = $(119304647)_{10}$
F2 = $(111000111000111000111000111000)_2$ = $(954437176)_{10}$
Verification:
F1 + F2 = B1 + B2 = $(1073741823)_{10}$ = $(111111111111111111111111111111)_2$
In L1 and L2, the ones reside in the leftmost positions (in senior bits) in relation to the groups of ones in F2 and F1 correspondingly:
L1 = $(100000100000100000100000100000)_2$ = $(1090785344)_{10}$
L2 = $(1000001000001000001000001000)_2$ = $(136348168)_{10}$
F1 and F2 enable to decantinate MB into odd MB1 and even MB2 parts.
We further subtract MB2 from L1 and MB1 from L2. If a certain pattern contains a ”one” (which defines the disalignment of the corresponding characters in str and pattern), the “technological” one in the corresponding senior bit is used in subtraction. In order to determine the remaining (non-impaired) ones, we apply a logical operation and a combination of the odd MB1 and the even MB2 parts. In the obtained statement MM = L1 AND (L1-MB2) + L2 AND (L2-MB1), ones reside in front (on the left, in the senior bit) with regard to the related segments. These ones mark (tag) the related segments. In order to obtain the MS disalignment pattern, it is necessary to fill the segments of the overlap and then to invert the bits. This can be done in the following way:
MS = (F1 + F2) – (MM – (MM -> 3))
Here, “-> n” refers to the right register shift (towards junior bits) by n positions, which is the basic operation of the processor.

Verification may be conducted by the direct comparison of str (bcbcbccfeg) and the pattern pattern (bcdabcbcbc). The set of overlapping characters (bc--bc----) corresponds to the obtained value of MS = $(111111000000111111000000000000)_2$.
Several peculiarities of LP approach application in TBS tasks algorithmization are worth noting.
A TBS task as a whole allows parallelizing. Moreover, parallelizing of comparison procedures application is practically the only permissible and acceptable resource of real-time mode provision in TBS tasks in connection with big data arrays. The difficulty lies in the fact that almost every single classical algorithm (fig. 1) was developed within the paradigm of character-by-character comparison. The word pattern is compared with the line block character by character. When disalignment of a symbol is found, comparison stops and the word pattern shifts to the new block. The given paradigm is stipulated (actually imposed) by the symbol representation (storage and processing) of meaningful information in a computer. Therefore, a symbol in classical TBS algorithms is the touchstone and the stumbling stone of the comparison procedure. Thus, within the framework of classical TBS algorithms, parallelizing may be effectuated by means of breaking the searched text into blocks and parallel launch of a separate part of the TBS algorithm for every text block on a separate processor. Thus, classical parallel (wide parallel) processing is meant.
In LP representation, the information is stored and processed not symbol by symbol, but in the form of LP multi-segment (multi-character) “integrations”. PR are compared with each other and, in this way, not separate symbols may overlap, but “integrations” of several symbols. Which symbol exactly did not coincide? Register shift procedure may be applied in order to define this. But then we start symbol manipulation and do not benefit from the application of LP. The requirement to the real LP realization of the TBS algorithm is to avoid symbol manipulation.
In BM algorithm, a displacement vector exists (is formed in compliance with certain rules) along with the word pattern. In comparison, the disalignment character position is found, the corresponding displacement component is taken and the word patters shifts along the text line with regard to the content ofthis component.
In the abovementioned segmental comparison algorithm, MS disalignment pattern is formed. It contains information about the location of the disaligned symbol. The key requirement to LP variant (analog) of the BM algorithm is not to convert this information into a numeral, but to directly form the necessary shift. Implementability of this procedure is defined by the fact that the numeral is an anthropomorphic (humanlike, made by the man or for the man) object. Conversion of certain acts with certain objects into certain numerals (i.e. representation by certain numerals) and further execution of these acts in compliance with these numerals (i.e. in compliance with the previously made representations) are a purely humanlike approach and a peculiarity of a certain stage of human intellect development. As a side note, the human intellect

along with BT media created functional-style programming, in which a series of intermediate symbols is absent. Thus, in Haskell programming language, the notions of a variable, a cycle (an, correspondingly, a cycle variable) are absent and the function of clipping (lambda function) may be used indirectly. The given backward-looking reference to approaches on the basis of functional-style programming may be regarded as a weighty argument in support of prospective viability and feasibility of search of PL variants (analogs) for BM algorithm as well as other (Fig.1) TBS algorithms.

## 5. CONCLUSION

Having analyzed LP information processing principles as applied to TBS tasks, the findings below were obtained:
1. Text block search algorithms classification was presented, the operation algorithm of which shows the possibility of their adaptation for LP implementation. The classification enables to distinguish common structural elements for program execution in the LP variant.
2. Comparative analysis was conducted for the advantages of LP approach application in TBS tasks, which enables to select the optimal alphabet size for text information representation by means of adjusting them to certain processor capacities.
3. Bit-by-bit and segmental LP word pattern processing options, which are the key procedures in text block search tasks, were proposed and exemplified. The given numerical values are applicable as test cases for program implementation.
4. Opportunities and difficulties of text block search algorithms LP realization were analyzed through the example of Boyer-Moore algorithm, which formed the grounds for the formulation of requirements and criteria for newly developed LP algorithms.

## REFERENCES


1. Barkovska O.Ju., Pyvovarova D.I., Serdechnyj V.S., Ljashova A.O. **Pryskorenyj alghorytm poshuku sliv-obraziv u teksti z adaptyvnoju dekompozycijeju vykhidnykh danykh (Advanced Algorithm of Word Patterns Search in Texts with Adaptive Output Decomposition).** // Systemy upravlinnja, navighaciji ta zv'jazku. – Poltava: PNTU, 2019. – Issue №. 4(56). – pp.28-34 https://doi.org/10.26906/SUNZ.2019.4.028
2. D. Minnie and S. Srinivasan, **"Intelligent Search Engine algorithms on indexing and searching of text documents using text representation,"** 2011 International Conference on Recent Trends in Information Systems, Kolkata, 2011, pp. 121-125. doi: 10.1109/ReTIS.2011.6146852.
3. Diachenko, V., Liashenko, O., Ibrahim, B.F., Mikhal, O., Koltun, Y. "**Kohonen network with parallel training: Operation structure and algorithm"** International Journal of Advanced Trends in Computer Science and Engineering, 8(1), pp. 35-38
4. Mikhal O.F. **Modelirovanie raspredelennykh informatsionno-upravlyayushchikh sistem sredstvami lokalno-parallelnykh algoritmov obrabotki nechetkoy informatsii (Modelling of Distributed Information Management Systems by Means of Local Parallel Fuzzy Information Processing Algorithms)** // Problemy bioniki. Vseukrainskiy mezhvedomstvennyy nauchno-tekhnicheskiy sbornik. Kharkov: KhNURE. 2001. Issue №54. pp. 28-34.
5. P. K. Chong, E. K. Karuppiah and K. K. Yong, **"A Multi-GPU Framework for In-Memory Text Data Analytics,"** 2013 27th International Conference on Advanced Information Networking and Applications Workshops, Barcelona, 2013, pp. 1411-1416. doi: 10.1109/WAINA.2013.238
6. **Data Science & Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data**/ EMC Education Services. David Dietrich, Barry Heller, Beibei Yang. Published by John Wiley & Sons. Inc. USA, 2015. 435 p.
7. E. A. Calvillo, A. Padilla, J. Muñoz, J. Ponce and J. T. Fernandez, **"Searching research papers using clustering and text mining,"** CONIELECOMP 2013, 23rd International Conference on Electronics, Communications and Computing, Cholula, 2013, pp. 78-81. doi: 10.1109/CONIELECOMP.2013.6525763
8. Christopher D. Manning, Prabhakar Raghavan, Heinrich Schütze. **Introduction to Information Retrieval: Translated from English.**– M.: JSC «I.D. Williams», 2011. – 528 p.
9. L. Diesendruck, L. Marini, R. Kooper, M. Kejriwal and K. McHenry, **"Abstract: Digitization and Search: A Non-Traditional Use of HPC,"** 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, Salt Lake City, UT, 2012, pp. 1460-1461. doi: 10.1109/SC.Companion.2012.259
10. Barkovska O., Serdechnyi V**. Control Model of Data Stream Transmitted over a Network Based on Proxying Technology.** Informatics Control Measurement in Economy and Environment Protection. – 2018. – Volume 8, No. 1. – P. 8-11.
11. Smelyakov, K., Sandrkin, D., Ruban, I., Martovytskyi, V., & Romanenkov, Y. (2018, October). **Search by image. New search engine service model.** In 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T) (pp. 181-186). IEEE. https://doi.org/10.1109/INFOCOMMST.2018.8632117