DE GRUYTER OPEN

THE EUROPEAN JOURNAL OF APPLIED ECONOMICS

EJAE 2015, 12(1): 52-57
ISSN 2406-2588
UDK: 004.424.4.056
DOI: 10.5937/EJAE12-8153
Original paper/Originalni naučni rad

APPROXIMATE SEARCH FOR BIG DATA WITH APPLICATIONS IN INFORMATION SECURITY – A SURVEY

Slobodan Petrović*
Gjøvik University College,
p.o. box 191, N-2802 Gjøvik, Norway

Abstract:
This paper is a survey of approximate search techniques in very large data sets (so-called Big Data). After a short introduction, some techniques for speeding up approximate search in such data sets based on exploitation of inherent bit-parallelism in computers are described. It then reviews the applications in search related to information security problems (digital forensics, malware detection, intrusion detection) are reviewed. Finally, the need for constraints in approximate search regarding the number of so-called elementary edit operations and the run lengths of particular elementary edit operations is explained and the status of on-going research on efficient implementation of approximate search algorithms with various constraints is given.

Key words:
Big Data,
approximate search,
bit-parallelism,
information security,
constraints.

INTRODUCTION

Big Data is a common name for the advanced techniques capable of analyzing very large data sets consisting of items different in form, stability, reliability *etc*. The data sources can be so large that even the theoretically efficient (sub-linear time complexity) algorithms cannot produce search results within a reasonable time period (exabytes of data). The form of the data can vary – it can be text, video, image, sound *etc*., and the relations among data items have to be found in such heterogeneous data sets. The velocity of data can be extreme (hundreds of gigabytes per second) and some measurements producing data can give unreliable results, thus reducing the quality of decision-making based on them. Such data must be discovered, replaced and/or approximated. In the area of business, data have been treated as asset for a long time already (Pries *et al*., 2015). When considering big data analytics, we usually talk about the so-called 5 Vs: Volume (up to exabytes of stable data to process), Velocity (huge quantities of data per unit of time and the response to them has to be given in an extremely short period of time), Variety (text, video, image, sound, structured, unstructured), Veracity (incomplete, inaccurate measurements, deception), and Value (business models can be related to data – possession or analysis of data) (Berre, 2015; Pries *et al*., 2015). Big Data has as a task to develop efficient algorithms for the analysis of such data.

As regards the static data volume, and in accordance with (IBM, 2013) 2.5 quintillion *(i.e.* 2.5×10^{18}) bytes of data are produced every day and 90% of all data has been produced in the last two years. The static volume by itself is no longer considered a problem even for such quantities of data. However, this is too optimistic, since static data in such analysis are treated as error-free, which is highly unrealistic. In order to analyze huge quantities of data (or queries) possibly containing errors, we need to employ the techniques of approximate search, which are much less efficient than those used in exact search. In such a situation, the static high-volume data processing becomes a problem again. When it comes to the implications

* E-mail: slobodanp@hig.no

of this fact to information security in digital forensics, we have to use approximate search techniques to cope with random or deliberately introduced errors in captured data and queries. Even though the time limits in digital forensic investigations are expressed in days and not in milliseconds, the amounts of captured static data are so large that using the existing approximate search techniques for forensic analysis becomes a problem.

As regards velocity, we usually consider analytics on streaming data (sound, video) (Berre, 2015), but in other applications, such as malware detection or intrusion detection, velocity matters as well. For instance, regarding network intrusion detection, even ordinary home networks deal with velocities of several gigabytes per second and very often the intrusion analysis has to be performed on group links, whose velocities reach hundreds of gigabytes per second (Eyrich, 2011; Sommer, 2011). In addition, new attacks and malware are derived from the existing ones and to avoid frequent change of signatures to detect them, we can use approximate search techniques to detect new attacks obtained by modifying the old ones. However, in order for the detection system to operate in real time, the approximate signature search algorithm must cope with huge data traffic velocities.

Variety of data has attracted much attention in big data analytics, since it is very interesting in business applications where we have to find relations among data from many heterogeneous sources (Berre, 2015; Pries, 2015). Variety of data matters in information security as well, especially in digital forensic investigations, where the analysts have to efficiently find relations among data in different forms captured from many sources. Based on such analysis, it is necessary to make the decision on whether or not to present the data as evidence in the court of law. Information is often deliberately changed by the suspect at the time of recording in the memory of the computer in order to make the eventual criminal investigation more difficult. Then, the approximate search techniques come again to the scene to deal with such deliberate data alterations, whose form can range from ordinary text to images and videos.

Veracity of data has to be taken into account in many applications, especially when we analyze sensor data. Nowadays, sensors are cheap and are deployed massively almost everywhere. With large machinery and transport facilities, sensors are used not only to estimate the current state of the equipment, but also to predict the future events. This phenomenon is called surprise detection (Frigessi, 2015). Similar approach can be used in information security in order for the security analyst to get a general idea about the monitored system, where some data can be missing or inaccurate (Gorodetsky *et al*., 2004). Afterwards, we need to compare data vectors where some features may be missing, which calls for approximate search in order to cope with such errors. The search must be very efficient since we need fine granularity of situation assessment in time.

In order to increase the efficiency of approximate search techniques, interdisciplinary approach is needed. Not only the new sophisticated search algorithms are invented for this purpose, but also advances in hardware technologies are combined with new software techniques (parallel approximate search algorithms and their simulations on conventional hardware (Layer, 2007; Navarro *et al*., 2002), multi-level logic, *etc*.), see Section 2. In addition, we often have to introduce constraints in the approximate search algorithm. As a result, new practically useful systems for big data analytics are designed. This also includes new efficient systems for forensic analysis and malware detection.

BIT PARALLELISM AND APPROXIMATE SEARCH

In this section, we shall concentrate on the so-called bit-parallelism phenomenon that has been extensively used in speeding up search algorithms in the past 20 years or so (Faro *et al*., 2012). We first present the definitions of Deterministic and Non-Deterministic Finite Automaton and then explain the bit-parallelism and its application in exact and approximate search.

A Deterministic Finite Automaton (DFA) is a Finite State Machine (FSM) with a property that from each state, given the same input symbol, there is only one transition to the next state. On the other hand, a Nondeterministic Finite Automaton (NFA) is a finite state machine capable of making transitions to more than one state for the same input symbol. One possible interpretation of such behavior is that the machine makes copies of itself in such a situation, which is then parallel processing. Each copy of the NFA processes the subsequent input symbols independently. If after performing such copy making and following one of the paths, an input symbol arrives, which does not appear as a label of any edge going out from the reached state on one of the machine copies, that machine-copy is stopped. It becomes inactive. If any of the copies of the machine reaches the final state, the input string is accepted, *i.e.* recognized.

Any NFA can be transformed to a DFA. The general transformation algorithm has exponential complexity with respect to the length of the string determining the NFA. However, we consider a simple special case, where this transformation can be performed in polynomial time. In this paper, we assign NFA and DFA to the search pattern, which is common in search applications. Then, the input sequence comes from the search string.

An NFA can be represented in two ways: with ε-transitions and without ε-transitions, where ε-transitions are transitions that do not consume any input. In both cases (with or without ε-transitions), such an NFA recognizes all the suffixes of the corresponding search pattern. Without ε-transitions, the NFA starts at the state 0 and remains active at least until the next symbol of the search string appears at the input (Fig. 1). Because of that, 0 is always an active state. If there is a path from the initial state to some state of the machine for a given input sequence of symbols, the final state of that path is called an inactive state. A state that is not active is called inactive state. A DFA can have only one active state at a time, whereas an NFA can have many such states concurrently.

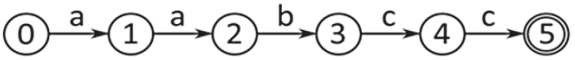

Figure 1. The NFA without e-transitions corresponding to the search pattern w=aabcc.

Each input symbol (*i.e.* the next symbol from the search string) drives creation of a new copy of the NFA, which starts at the state 0.

With ε-transitions, the active states of such a machine are those that correspond to ending symbols of input sequences. For example, consider the search pattern from Fig. 1, w=aabcc. The corresponding NFA with ε-transitions is given in Fig. 2. If the input sequence is "aa", then the state 2 of the machine will be active after receiving this sequence. If the input sequence is "c", the states 4 and 5 will be active. If the input sequence is "aba", no state is active and the whole machine is considered inactive.

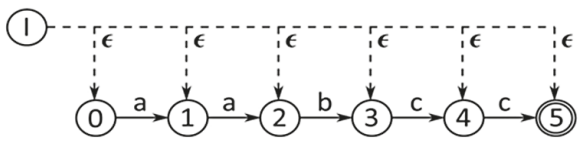

Figure 2. The NFA with e-transitions corresponding to the search pattern w=aabcc.

Since an NFA of this kind (with or without ε-transitions) recognizes all the suffixes of the search pattern w, the corresponding DFA is so-called Directed Acyclic Word Graph (DAWG) or suffix automaton of *w* (Fig. 3) and can be obtained by means of an algorithm with polynomial time complexity (Navarro *et al.*, 2002).

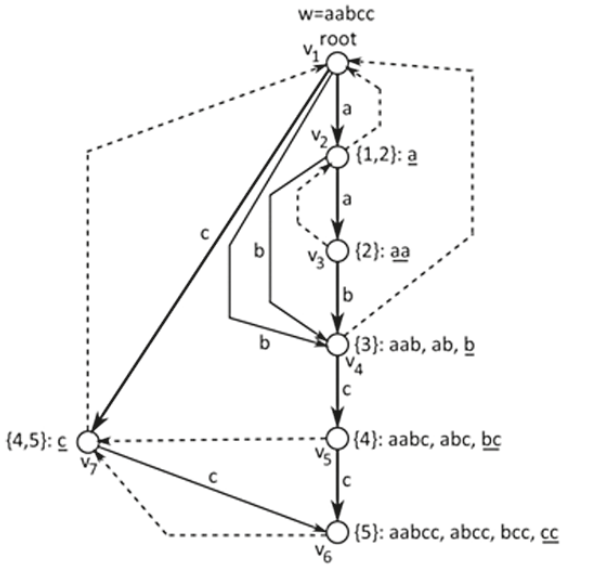

Figure 3. A Directed Acyclic Word Graph (DAWG) corresponding to the pattern w=aabcc.

We explain the bit-parallelism phenomenon on an example. Suppose that we are given the search string S=aaabcaabcc and we are searching for the pattern *w*=aabcc in it by means of the NFA without ε-transitions. Each time a symbol from *S* arrives, the machine makes a copy of itself and starts from the 0 state. Suppose the maximum number of machines running in parallel is $m=|w|$, in our example $m=5$. This means that we simulate parallel processing on a real computer instead of theoretical consideration of unlimited parallelism. We denote by *j* the number of processed symbols from *S*. Then, we have min (j,m) machines running in parallel, for each *j*. After processing *j* symbols from *S*, some of these machines are active and some are inactive.

Define search status in a computer word D of m bits. In our case, $D=d_5d_4d_3d_2d_1$. We set $d_i=1$ if the corresponding machine is active after processing *j* bits of *S*. Before processing any symbol, all the machines are active (since they are all in the state 0) and consequently $D=1^m$ (all ones). The operation of such an NFA is illustrated in Fig. 4.

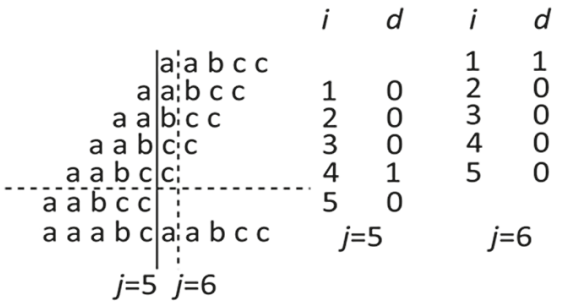

Figure 4. Operation of an NFA (see text).

As can be seen in Fig. 4, when passing from *j=5* to *j=6*, the bit d_5 disappears, since we can have maximum m machines at a time. This fact is expressed by shifting the search status word *D* one position to the left (d_4 becomes d_5, d_3 becomes d_4, *etc.*) At the same time, a new machine is created corresponding to the bit d_1, which starts from the state 0 (which is always active). This fact is expressed by OR-ing the shifted word *D* to the left with $0^{m-1}1$ (in our case *m=5* so we OR with 00001).

In our example, when passing from *j=5* to *j=6*, the next symbol from *S* to be processed is a. If we ask ourselves which input symbol will keep which machine active, if it was active before processing that symbol, the answer in this particular case is that an a will always keep the machine d_1 active, an a will keep the machine d_2 active, a b will keep the machine d_3 active, *a c* will keep the machine d_4 active, and *a c* will keep the machine d_5 active. We can use this fact for updating the search status word *D* automatically, after each shift and OR-ing with 1, by introducing the *bit masks*.

The bit mask for any symbol depends only on the search pattern, not on the search string. Because of that, we can pre-compute the bit masks for all the symbols from the pattern *w*. Given the search pattern $w=w_1w_2\ldots w_m$, for the bit mask $B[s]=b_1b_2\ldots b_m$ the following holds: if $s=w_i$ then $b_{m+1-i}=1$, otherwise $b_{m+1-i}=0$.

In our example, since *w*=aabcc, it is easy to see that *B*[a]=00011, *B*[b]=00100, and *B*[c]=11000. We can now update the search status word *D* for each new input symbol S_j in the following way

$$D_j=((D_{j-1}<<1) \text{ OR } 1) \text{ AND } B[S_j] \quad (1)$$

In our example, for *j=6*, we have

$$D_6=((D_5<<1) \text{ OR } 1) \text{ AND } B[S_6]=$$
$$= ((01000<<1) \text{ OR } 00001) \text{ AND } B[\text{a}] =$$
$$= 10001 \text{ AND } 00011 = 00001.$$

The corresponding exact search algorithm is called Shift-AND and was the first to appear back in 1989 (Baeza-Yates *et al.*, 1989). By complementing the bit masks and the search status word *D*, we obtain a bit more efficient implementation of that algorithm called Shift-OR. In that algorithm, the status word update formula is

$$D_j=(D_{j-1}<<1) \text{ OR } B[S_j] \quad (2)$$

Following the above-explained bit-parallelism implementation principle, the fastest known exact search algorithms on average, Backward Non-deterministic DAWG Matching (BNDM) (Navarro *et al.*, 2002) and BNDMq (Durian *et al.*, 2009) have been devised.

We now show how bit-parallelism techniques can be used in approximate search. The main idea is to design an NFA capable of encompassing error processing and then construction of a bit-parallel simulation of such an automaton is straightforward.

The NFA assigned to a search pattern with errors is presented in Fig. 5 (Navarro *et al.*, 2002). In Fig. 5, where a match is represented with a horizontal solid line (we advance 1 character in the search pattern and in the search text), an insertion with a vertical solid line (we advance 1 character in the search text but we do not advance in the search pattern), a substitution that is not a match with a diagonal solid line (we advance 1 character in the search pattern and in the search text) and a deletion with a diagonal dashed line (we advance 1 character in the search pattern but we do not advance in the search text) - this is an ε-transition. A loop in the state 0 of the NFA is used to represent waiting for the first character of the search pattern in the search string. If the NFA manages to find itself in one of the right-most states (double circled), the input string is accepted, which means that we have found the search pattern in the search string.

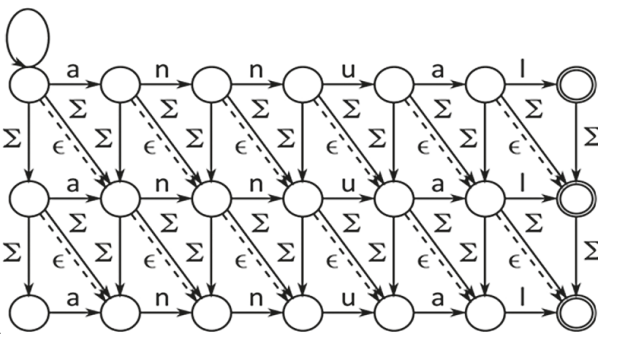

Figure 5. An NFA assigned to the search pattern w=annual with up to 2 errors.

The NFA simulation approach to approximate search introduced in Section 2 proved itself efficient. There are several ways to apply bit-parallelism in this simulation, of which the most important ones are Row-wise bit parallelism (RBP) and Diagonal-wise bit parallelism (DBP) (Navarro *et al*., 2002). The idea of RBP is to treat each row of the NFA given in the matrix form (see Fig. 5) as a separate NFA and for each of these NFAs we maintain a separate search status word. With the DBP approach we obtain a more complicated algorithm for implementation, but the worst-case complexity of the DBP approach is k times better than with the RBP approach, where k is the number of allowed errors in search. A k-error tolerant BNDM algorithm also exists (Navarro *et al*., 2002). Special modifications of all these algorithms are needed if wildcard search, regular expression matching or multi-pattern search is needed.

APPLICATIONS IN INFORMATION SECURITY

In information security, search is used for:

- Finding malware (known-to-be-malicious files).
- Finding evidence in criminal cases:
  - documents,
  - images,
  - sound, *etc*.
- Intrusion detection in real time, *etc*.

Search in digital forensics must be efficient since there is short time available to collect evidence and the captured data sets can be huge and heterogeneous. For real time systems, such as malware detectors and Intrusion Detection Systems (IDS), efficiency is essential since otherwise the attacks/malware would pass unnoticed, which produces the so-called *false negatives* in detection.

The search problem has been extensively studied in the last 50 years and several hundreds of (good) search algorithms are known. The efficiency of approximate search is a big challenge since it turns out that it is much more difficult to achieve than the efficiency of exact search. In addition, approximate search must quite often include various constraints (for example, on the total number of so-called edit operations that transform one string into another), which enables a more flexible approximate search, but it makes the search algorithms less efficient if not implemented carefully.

Multi-pattern approximate search is especially interesting in intrusion detection since we rarely have to search for just one pattern in the traffic – most attack signatures consist of several patterns.

In order to explain approximate search with constraints, we need the definition of the so-called edit distance. Let w be a search pattern and let S be a search string. Approximate search for w in S with k-error tolerance can be considered as search for a substring of S at the distance to w less than or equal to k. The distance measure used most often is edit distance (Levenshtein, 1966), which is defined as the minimum number of so-called elementary edit operations (deletions, insertions and substitutions) necessary for transforming one string into another. Edit distance is widely used since it can be computed in an iterative way, by filling a matrix of partial edit distances. Each entry in this matrix contains the edit distance of the corresponding prefix of one string to the corresponding prefix of another string. If we initialize this matrix in a special way (the first row is zeroed), we can run the algorithm for computation of edit distance to find w in S at the distance less than or equal to k (Navarro, 2002). It turns out that this kind of computation is equivalent to simulation of NFA in the matrix form described in the previous section.

In some applications (cryptanalysis of irregularly clocked Linear Feedback Shift Registers (LFSR) (Petrovic *et al*., 2007), matching of DNA sequences (Naor *et al*., 1993)), it is necessary to introduce constraints in the definition of edit distance. The constraints can be on the total number of elementary edit operations or on the run lengths of certain elementary edit operations (deletions and/or insertions) (Sankoff *et al*., 2000). In addition, it is sometimes necessary to reconstruct the edit sequences (the optimal sequences of edit operations that transform one string into another) in an efficient way. Other applications of constrained edit distance in information security are possible as well. Error-tolerant digital forensic search should also be considered. By introducing various constraints, we can recognize the human and/or the device that produced deliberate alteration of incriminating content of the captured device(s).

Since the time complexity of computation of both constrained and unconstrained edit distance and the reconstruction of the edit sequence is quadratic in the length of the search string, it would be of interest to speed up this by simulating NFA. It has been done for unconstrained edit distance (see Section 2), but for constrained edit distance it still needs to be accomplished. There are certain difficulties related to the simulation of the operation of the NFA in the constrained context. Namely, it is necessary to track the lengths of the paths leading to every active state of the NFA, which calls for introduction of additional memory into the device. In spite of this additional memory cost, the overall efficiency of the constrained edit distance computation will be improved.

SUMMARY

In this paper, a survey of techniques used to speed up search in large data sets by means of simulating Nondeterministic Finite Automata (NFA) is given. It shows how these techniques can be applied in information security, as well as how introducing various constraints in the definition of the distance used in error tolerant search influences the efficiency of the search algorithms of this kind, and information is provided on the current status of the research in compensating this influence.

REFERENCES

Baeza-Yates, R., & Gonnet, G.H. (1992). A New Approach to Text Searching. *Communications of the ACM*. 35(10), 74-82. DOI: 10.1145/135239.135243

Berre, A.J. (2015). *BigData Value PPP in Horizon 2020*. Retrieved March 30, 2015, from http://www.sintef.no/globalassets/sintef-ikt/arrangementer/2015-02-23_workshop_big_data/2---bdva-ppp-2402-berre.pdf

Durian, B., Holub, J., Peltola, H., & Tarhio, J. (2009). Tuning BNDM with q-Grams. In J. Hershberger & I. Finocchi (*Ed.*) *2009 Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments (ALENEX)*. DOI: 10.1137/1.9781611972894.3

Eyrich, J. (2011). *Drinking from a Firehose: How to get traffic to your Bro cluster?* Retrieved March 30, 2015, from https://www.bro.org/bro-workshop-2011/slides/drinking-from-fire-hose.pdf

Faro, S., & Lecroq, T. (2012). *Twenty Years of Bit-Parallelism in String Matching.* In J. Holub, B.W. Watson & J. Zdarek (Ed.) Festschrift for Borivoj Melichar (pp. 72-101). Prague: Czech Technical University.

Frigessi, A. (2015). *Big Insight*. Retrieved March 30, 2015, from http://www.sintef.no/globalassets/sintef-ikt/arrangementer/2015-02-23_workshop_big_data/4---biginsight_frigessi.pdf

Gorodetsky, V., Karsaev, O., & Samoilov, V. (2004). *On-Line Update of Situation Assessment Based on Asynchronous Data Streams*. In Knowledge-Based Intelligent Information and Engineering Systems: 8th International Conference, KES 2004 (pp. 1136-1142). Berlin: Springer Verlag. DOI: 10.1007/978-3-540-30132-5_154

IBM. (2015). *What is big data?* Retrieved March 30, 2015, from http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

Layer, C. (2007). *A Coprocessor for Fast Searching in Large Databases: Associative Computing Engine*. PhD thesis, University of Ulm.

Levenshtein, V. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physical Doklady*, 10(8), 707-710.

Naor, D., & Brutlag, D. (1993). *On Suboptimal Alignments in Biological Sequences.* In A. Apostolico *et al.* (Ed.) Combinatorial pattern matching: 4th annual symposium, CPM 93, Padova, Italy, June 2-4, 1993 : Proceedings (pp. 179-196). Berlin: Springer Verlag.

Navarro, G., & Raffinot, M. (2002). F*lexible Pattern Matching in Strings: Practical on-line search algorithms for texts and biological sequences.* Cambridge: Cambridge University Press.

Petrovic, S., & Fuster-Sabater, A. (2007). *Reconstruction of Suboptimal Paths in the Constrained Edit Distance Array with Application in Cryptanalysis.* In O. Gervasi & M. L. Gavrilova (Ed.) Computational Science and Its Applications - ICCSA 2007: International Conference, Kuala Lumpur, Malaysia, August 26-29, 2007: Proceedings, Part III (pp. 597-610). Berlin: Springer Verlag. DOI: 10.1007/978-3-540-74484-9_52

Pries, K.H., & Dunnigan, R. (2015). *Big Data Analytics: A practical guide for managers*. Boca Raton: CRC Press.

Sankoff, D., & Kruskal, J. (2000). *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*. Cambridge: Cambridge University Press.

Sommer, R. (2011). *Broverview*. Retrieved March 30, 2015, from https://www.bro.org/bro-workshop-2011/slides/broverview.pdf

APROKSIMATIVNE PRETRAGE U VELIKIM SKUPOVIMA PODATAKA SA PRIMENAMA U BEZBEDNOSTI INFORMACIONIH SISTEMA - PREGLED

Rezime:

Ovaj rad nudi pregled tehnika aproksimativne pretrage u velikim skupovima podataka ("Big Data"). Nakon kratkog uvoda, prikazane su neke od tehnika za ubrzanu aproksimativnu pretragu u takvim skupovima podataka zasnovanih na ispitivanju inherentnog bit-paralelizma u računarima. Razmatraju se i njihove primene vezano za probleme bezbednosti informacionih sistema (digitalna forenzika, detekcija malvera, detekcija upada). Na kraju se objašnjava potreba za ograničenjima u aproksimativnim pretragama vezano za broj takozvanih osnovnih operacija za editovanje i dužinu trajanja istih, kao i trenutno stanje istraživanja o efikasnom sprovođenju algoritama aproksimativne pretrage sa različitim ograničenjima.

Ključne reči:

veliki skupovi podataka,
aproksimativne pretrage,
bit-paralelizam,
bezbednost informacionih sistema,
ograničenja.

Received: February 22, 2015.
Correction: March 16, 2015.
Accepted: April 5, 2015.