A New Approach to Hash Function Construction for Textual Data: Comparison,
IEEE WICT 2014 Conference, pp.39-44, ISBN 978-1-4799-8115-1, Malaysia, 2014

A New Approach to Hash Function Construction for Textual Data: A Comparison

Vaclav Skala, Radek Petruska
Department of Computer Science and Engineering
University of West Bohemia
Plzen, Czech Republic
http://www.VaclavSkala.eu

***Abstract*—Many techniques for text processing are based on efficient data storing and retrieval techniques. Careful selection of data structures used and retrieval techniques play a significant role in efficiency of the whole system of data processing. Hashing technique is one very often used technique with O(1) run-time complexity for data storing and retrieval. A comparison of new technique for hash function construction is presented in the paper without need of division operation. The comparison of the proposed technique is especially convenient for large textual data sets processing. State of the art in hashing of textual data is given (the perfect hashing techniques are not included). The proposed hash function construction and hashing technique have been compared with other comparative techniques for different languages and textual data (chemical data sets etc.).**

Keywords-Hashing function, information retrieval, text processing, text mining, summarization, large data processing, data structure

I. INTRODUCTION

Many problems require fast determination whether the given item, textual, graphical or geometrical, is already stored in the dataset. Resolving this problem can be very difficult, especially as the size of the data set increases. A typical example of application is duplicity elimination, e.g. in textual data sets. One technique convenient for solution of this problem is an application of hashing data structure. The advantage of the hash data structure is that data storage and retrieval is of $O(1)$ run-time complexity if the hash function is well designed or if a perfect hash function is used.

Table 1: Differences between textual and geometrical data

Interval of values	Dimensionality	
	Small	High
Small	Image data Dim 2,3 Values $\langle 0{,}255 \rangle$	Textual data Dim (string length) Values $\langle 0{,}255 \rangle$ (ASCII etc.)
High	Geometrical Dim 2,3 Values $(-\infty, +\infty)$	Harmonic analysis

It can be seen that there is a significant difference between textual and geometrical data. However the hashing principle is common, but only the hash function is constructed differently. The main differences are:

- *Textual data* – interval of values is given by alphabet used, nevertheless the dimensionality is high as strings might be very long, e.g.:
  - protein *titin* is decribed by 189,819 characters
  - name of the railway station in Wales *Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch*

- *Geometrical data* – the dimensionality is usually 2 or 3 as points are represented by coordinates $\langle x, y \rangle$, resp. $\langle x, y, z \rangle$, but interval of values is "unlimited" $(-\infty, +\infty)$ and high number of items are processed, typically $10^6 - 10^{14}$ points etc.

There are two approaches to the hash function design:

1. *The perfect hash function* design is applicable to the final data sets that are not expected to change, and its computation is of *O(M)* expected complexity for the given data set [12]. The perfect hash function gives a unique index for each item from the data set. *The minimal perfect hash function* is the perfect hash function for which the hash table has no holes, i.e. the size of the hash table is equal to the number of items. This hash function can be made for a static list only and it is usually referred to as *the dictionary problem* [5].
2. The hash function design described in this paper is based on experience with recently designed hash functions. Such an approach must be used in the case where the hash table is build incrementally. However some problems will occur:
   - To design a hash function properly, the fundamental requirement is that the number of collisions must be as small as possible. Collision occurs when different items are transformed to the same index to the hash table.
   - There can be a problem with memory requirements as the size of the hash table rises, particularly as the functionality of the hash function depends on the hash table length.

   Hash functions have been also used effectively in several geometrical applications [6], for the duplicate elimination among geometric entities. The experiments with geometric applications made recently [24] led to a question whether a similar approach can be taken for string-based problems as well, especially for large data sets and for batch and incremental processing as well. This resulted to a new approach for textual data processing.

Table 1: Complexity of approaches to duplicity elimination

	Batch processing complexity (for all items)	Insert one item with duplicate elimination
Sort and duplicate elimination	*O(M lgM)*	*O(N)*
Using a tree including balancing	*O(M lgM)*	*O(lgN)* without balancing
Hash function use (expected)	$O(M*I_a)$	$O(I_a)$
Hash function use (worst case[1])	$O(N^2)$	*O(N)*

where: M is the number of items, I_a, I_m is the average, resp. the maximal cluster length.

II. Hash Function and Data Structure

Principle of the hashing is very simple. It is actually based on indirect addressing principle. Processed element x is transformed to an address $addr$ which points to a table TAB, where the location of the element x is stored, see Fig.1.

Standard function use $\boldsymbol{mod}$ operation and generally it has the following form:

$$h(x) = \left(\sum_{i=1}^{L_x} x_i \right) \boldsymbol{mod}\ p \qquad (1)$$

where p is a prime number, L_x is the length of the given string, x_i. is the i-th character of the given string.

The proposed S-Hash (Smart Hash) function is based on floating point operations instead of integer and instead of $\boldsymbol{mod}$ operation, masking and shifting is used. The S-Hash function is constructed as follows:

$$h(x) = \left\lfloor c \sum_{i=1}^{N} x_i q^i \right\rfloor \boldsymbol{and}\ (HS - 1) \qquad (2)$$

$$c = \frac{(2^m - 1)(1 - q)}{L_{max}} \qquad (3)$$

where HS is the hash table length, i.e. $HS = 2^{\lceil log_2 \frac{1}{f} N \rceil}$, f is the load factor. i.e. $f \geq 2$, c is a hardware dependent constant, $m = 64$ for 64 bits platform, $0 < q < 1$ is "irrational", i.e. $1/n$, where n is an integer, e.g. 1/3 etc. (unlimited fractional part).

The S-Hash has function construction has several advantages, namely:

- hash function does not need division operation by a prime, instead logical operation $\boldsymbol{and}$ is used, if has table has to be shorten,
- no re-computation if the VAL data structure is needed, the TAB table of the length 2^k can be simply "recomputed" by folding upper and lower parts of the lengths $2^k - 1$.

Structures TAB and VAL can be stored in parts, i.e. distributed processing is supported in the case of very large data sets.

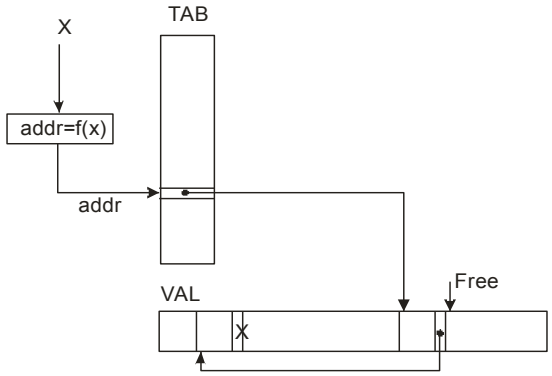

Figure 1: Principle of dynamic memory management based of hashing data structure

III. Comparison Criteria

To be able to compare different hash functions it is necessary to introduce some general criteria. Assume that there are already *N* items stored in the data structure and *I* is the cluster length. Three basic situations can occur when a new item i.e. a string is inserted to the structure:

1. The item is not stored in the data structure and the appropriate cluster is empty. The item is inserted to this cluster. The cost of this operation for all such items can be expressed as:
$$Q_1 = 0 \qquad (4)$$
2. The item is not stored in the data structure so the whole cluster is to be searched and the item is to be inserted to an appropriate cluster. The cost of this operation for all such items can be expressed as (because the cluster of the length *I* must be searched for all items in this cluster *I*-times, value *I* is powered by two)
$$Q_2 = \sum_{I=0}^{I_n} I^2 C_I \qquad (5)$$
3. The item is stored already in the data structure so the corresponding cluster is to be searched and the item is not inserted to the appropriate cluster. Because only half of the cluster is to be searched on average, the cost of this operation for all such items can be expressed as
$$Q_3 = \sum_{I=0}^{I_n} \frac{1}{2} I^2 C_I \qquad (6)$$

It is necessary to point out that the cost of the hash function evaluation has not been considered, as it is the same for all cases. The cost of item insertion to a cluster was omitted. The final criterion can be expressed as

$$Q = Q_1 + Q_2 + Q_3 = \sum_{I=0}^{I_n} \frac{3}{2} I^2 C_I \qquad (7)$$

Empty clusters are not considered by this criterion because the hash table length *HS* depends on the number of items stored. It can be seen that the criterion Q depends on the number of items. We used a relative criterion to evaluate properties of hash functions for different data sets with different sizes defined as

$$Q' = \frac{Q}{N} \tag{8}$$

IV. S-HASH FUNCTION PROPERTIES

The proposed S-Hash function was recently tested for textual and geometrical data as well in order to prove expected properties. For illustration, how the cluster length depends on a parameter q for the Czech and English languages see Fig.2 and Fig.3. Peaks occur when the q value is not "irrational".

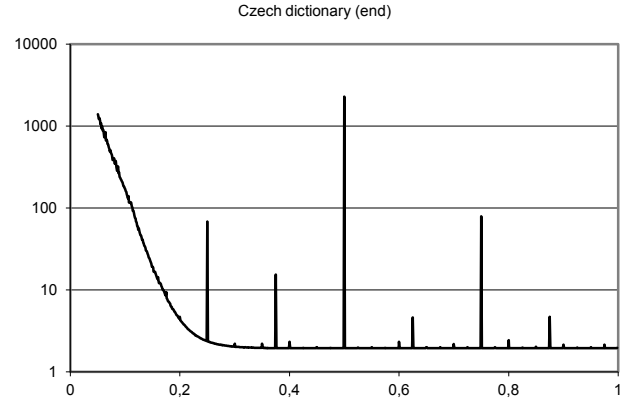

Figure 2: Relative criterion for Czech dictionary

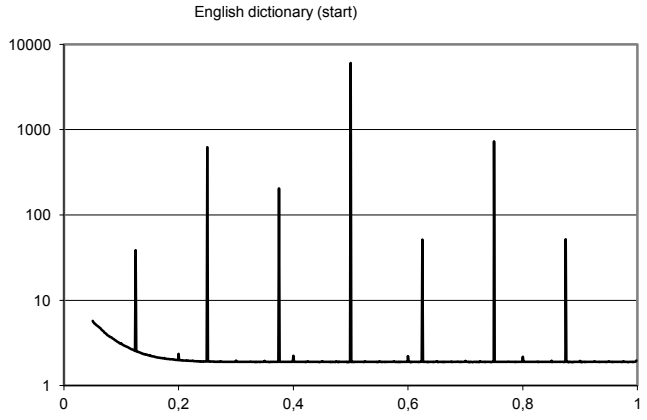

Figure 3: Relative criterion for English dictionary

It can be seen that there are some small differences due to different "language structure". As S-Hashing behavior was good a comparative study was made.

V. COMPARATIVE EXPERIMENTAL RESULTS

The proposed S-Hash function was recently compared with the main hashing functions used nowadays, i.e. AP (Arash Partow), BKDR (Brian Kernighanm, Dennis Ritchie), DJB (Dan J.Bernstein), ELF, FNV, Java, Rotational SDBM.

For the comparison the following databases were used: ECHA (European Chemicals Agency), ChEBI (Chemical Entities of Biological Interest), NIST (WebBook Chemie), PDB (Protein database bank) and EkoTox (Ecotoxilogic database) and two dictionaries, Czech and English, were used [9]. Selected hash functions were tested using different textual data bases, i.e. text with natural languages, i.e. French, German, English, Russian, Hebrew, specialized texts, like chemicals etc. Fig.4 presents typical values for bucket length evaluation and Fig.5 presents ratio of bucket lengths for selected methods against proposed S-Hash function.

For evaluation also a linear and quadratic length cluster average was used

$$I_a = N/M \qquad I_{2a} = \sqrt{\sum_{i=1}^{l_m} I^2 C_l \Big/ M} \tag{9}$$

The following experimental results have been obtained for different databases.

Due to recent experiments the parameter $q \in \langle 0.9, 1)$, for complex chemical compounds $q \in \langle 0.8, 0.9)$ was used. Criterion Comp. is defined as $Q' = {}^{Q}/_{N}$ in order to make results independent of number of items peocessed.

ECHA Database

Agency ECHA (European Chemicals Agency) is one of regulatory EU institution responsible for safety use of chemicals. List contains 6 500 records.

Table 2: ECHA database

Funkce	M-B	Q'	Comp. Q'	I_a	I_{2a}
Shash	6	2,531	0,000	1,39	1,53
AP	5	2,564	0,034	1,40	1,55
Java	6	2,577	0,046	1,41	1,55
Rotační	5	2,590	0,060	1,41	1,56
ELF	6	2,599	0,069	1,42	1,57
DJB	6	2,600	0,069	1,41	1,56
SDBM	6	2,603	0,072	1,41	1,57
FNV	6	2,611	0,080	1,42	1,57
BKDR	6	2,625	0,094	1,42	1,57
DEK	9	2,907	0,376	1,48	1,69
Aditivní	8	2,968	0,437	1,52	1,74
XOR	65	70,325	67,795	34,17	40,02

ChEBI database

ChEBI (Chemical Entities of Biological Interest) database contains approx 38 000 records of organic compounds.

Table 3: ChEBI database

Function	M-B	Q'	Comp. Q'	I_a	I_{2a}
Shash	5	2,357	0,000	1,32	1,44
FNV	6	2,381	0,025	1,32	1,45
AP	6	2,391	0,035	1,33	1,45
BKDR	6	2,393	0,036	1,33	1,45
DJB	6	2,393	0,036	1,33	1,45
SDBM	6	2,394	0,037	1,33	1,46
Java	7	2,396	0,039	1,33	1,46
ELF	6	2,408	0,052	1,33	1,46
Rotační	7	2,448	0,091	1,34	1,48
DEK	25	3,334	0,978	1,52	1,84
Aditivní	38	18,616	16,260	5,47	8,24
XOR	367	460,030	457,673	303,98	305,33

PDB database

PDB (Protein Data Bank) is the protein database containing 15 000 records.

Table 5: PDB database

Function	M-B	Q'	Comp. Q'	I_a	I_{2a}
Shash	7	2,786	0,000	1,495	1,666
BKDR	6	2,818	0,032	1,509	1,684
FNV	7	2,834	0,048	1,507	1,687
Java	7	2,836	0,050	1,514	1,692
ELF	7	2,839	0,053	1,508	1,689
AP	6	2,839	0,053	1,508	1,690
Rotating	7	2,843	0,057	1,510	1,692
DJB	7	2,846	0,061	1,515	1,695
SDBM	7	2,851	0,065	1,515	1,697
DEK	7	2,978	0,192	1,548	1,753
Additing	8	3,636	0,850	1,815	2,098
XOR	145	173,147	170,361	114,383	114,906

NIST Database

NIST (National Institute of Standards and Technology) database contains approx. 72 000 records.

Table 4: NIST database

Function	M-B	Q'	Comp. Q'	I_a	I_{2a}
Shash	7	2,307	0,000	1,29	1,41
SDBM	6	2,324	0,017	1,30	1,42
AP	7	2,326	0,019	1,30	1,42
DJB	6	2,327	0,020	1,30	1,42
BKDR	6	2,328	0,021	1,30	1,42
Rotační	7	2,329	0,022	1,30	1,42
Java	6	2,332	0,025	1,30	1,42
FNV	7	2,340	0,033	1,31	1,43
ELF	77	2,537	0,230	1,31	1,49
DEK	70	3,286	0,979	1,38	1,74
Aditivní	38	23,153	20,846	9,48	12,10
XOR	2078	922,762	920,455	567,06	590,63

EkoTox database

EkoTox database contains compounds with toxicological hazards with selected substances with 200 000 records.

Table 3: EkoTox database

Function	M-B	Q'	Comp. Q'	I_a	I_{2a}
Shash	8	2,6699	0,0000	1,439	1,600
FNV	7	2,6728	0,0029	1,442	1,603
BKDR	8	2,6730	0,0031	1,442	1,603
AP	8	2,6752	0,0052	1,443	1,604
Java	6	2,6762	0,0063	1,443	1,604
DJB	7	2,6765	0,0065	1,443	1,604
Rotating	7	2,6774	0,0074	1,444	1,605
ELF	7	2,6790	0,0091	1,444	1,606
SDBM	7	2,6848	0,0148	1,446	1,609
DEK	25	3,0516	0,3817	1,524	1,761
Additing	85	49,5357	46,8657	12,419	20,251
XOR	1588	2046,227	2043,557	337,225	678,252

VI. CONCLUSION

This paper presents a comparison of S-Hash hashing methods for textual data. The S-Hasing method offers a common approach to textual and geometrical data. The behavior of the S-Hash function has been tested on Czech and English dictionaries as these two languages belong to different language groups and on different databases including chemical and toxicological databases.

For the proposed data structure the optimal hash table length was derived and also the recommendations for q values were verified. It was proved that proposed S-Hashing offers good computational properties and no division operations with primes is needed.

The influence of hash table length was experimentally verified for large data sests. It is clear that on computers with less memory swapping can be used, i.e. where some parts of the structure are stored on disk during the building of the hash table. However, the shorter hash table can be easily constructed without need of all data processing, if shortaen by a factor $1/2$ or $1/4$ etc. Of course, the bucket length will become longer.

ACKNOWLEDGMENT

The authors would like to thank to all who contributed to this work, especially to colleagues at the University of West Bohemia in Plzen who have stimulated this work.

Thanks belong also to anonymous reviewers for their critical comments that helped to improve this manuscript significantly. EkoTox database data courtesy of Dr.Pavel Pavliček and Vilém Čermák.

This project was supported by the MSMT CR Projects No.LH12181, LG13047 and SGS-2013-029.

REFERENCES

[1] Dzysyak, S.: Javascript hash functions to convert string into integer hash. *Erly Coder .com.* [Download: 8.4.2013] http://erlycoder.com/49/javascript-hash-functions-to-convert-string-into-integer-hash, 2011.

[2] *ECH:* European Chemicals Agency. [Download: 18.2.2013] http://echa.europa.eu/.

[3] Feng, Zukang a jiní. Ligand Expo. *Protien Data Bank.* [Download: 18.2.2013] http://ligand-expo.rcsb.org/, 2004.

[4] Fowler,G., Vo,P.: *FNV Hash*. [Download: 8.4.2013] http://www.isthe.com/chongo/tech/comp/fnv, 1994

[5] Gettys,T.: Generating perfect hash function, *Dr.Dobb's Journal,* Vol.26(2), 151-155, 2001.

[6] Glassner,A: Building Vertex Normals from an Unstructured Polygon List, *Graphics Gems*, IV, 60 - 73. Academic Press, Inc., Cambridge, 1994

[7] Guard, Damien. Calculating Elf-32 in C# and .NET. [Download:8.4.2013], 2007http://damieng.com/blog/2007/11/24/calculating-elf-32-in-c-and-net.

[8] ChEBI. *Chemical entities of biological interest.* [Download: 18.2.2013] http://www.ebi.ac.uk/chebi, 2009

[9] ISPELL: SPELL Dictionaries, http://ficus-www.cs.ucla.edu/geoff/ispell-dictionaries.html

[10] Knuth,D.I.: *The Art Of Computer Programming: Sorting and Searching*. 2nd edition. Addison-Wesley Profesional, 1998.

[11] de Matos, P., Alcántara, R., Dekker, A., Ennis, M., Hastings, J., Haug, K., a další. (2009). *ChEBI*. Chemical entities of biological interest: http://www.ebi.ac.uk/chebi/, Download 18.2.2013, 2009

[12] Dzysyak, S. (2011). *Javascript hash functions to convert string into integer hash*. Download 8.4.2013, Erly Coder.com: http://erlycoder.com/49/javascript-hash-functions-to-convert-string-into-integer-hash

[13] Engelschall, R. PASTEBIN: http://pastebin.com/dDQ2kDkK, Download 18.4.2013, 2012

[14] European Chemicals Agency, Download 18.2.2013, ECHA: http://echa.europa.eu/

[15] Feng, Z., Chen, L., Maddula, H., Akcan, O., Oughtred, R., Berman, H. M. et al. *Ligand Expo*., Protien Data Bank: http://ligand-expo.rcsb.org/, Download 18.2. 2013, 2004

[16] Fowler, G., & Vo, P. (1994)., FNV Hash: http://www.isthe.com/chongo/tech/comp/fnv/, Download 8.4.2013, 1994

[17] Knuth, D. E. *The Art Of Computer Programming.* Massachusetts: Addison-Wesley Profesional, 1998.

[18] Matouš, J., & Šípek, M. NIST WebBook Chemie: http://webbook.nist.gov/chemistry/ Download 18.2. 2013, 2009

[19] Mička, P. (2008). *Hashovací tabulka - Algoritmy.net*. Získáno 1. 4 2013, Algoritmy.net: (in Czech) http://www.algoritmy.net/article/32077/Hashovaci-tabulka

[20] Mulvey, B. *Pluto Scarab - Hash Functions*. Hash Functions: http://home.comcast.net/~bretm/hash/, Download 1.4.2013, 2007

[21] Partow, A. General Purpose Hash Function Algorithms, , http://www.partow.net/programming/hashfunctions/, Download 8.4.2013

[22] Pískač, P., & Čermák, V. Ekotoxikologická databáze: http://www.piskac.cz/ETD/Default.htm, Download 20. 2 2013, 1996

[23] Skala, V., Hrádek, J. *Effecient Hash Function for Duplicate Elimination in Dictionaries.* Bratislava: Slovak University of Technology, 2009.

[24] Skala,V., Kuchar,M.:The Hash Function and Principle of Duality, *IEEE CGI proceedings*, pp. 167-174, 2001, Hong Kong, 2001

APPENDIX

Fig.4 presents experimental results of the comparison of the S-Hashing technique for different databases. The proposed S-Hashing is slightly better than the other methods used in this comparative study. Fig.5 presents differential graphs, where S-Hash technique was taken as the reference method.

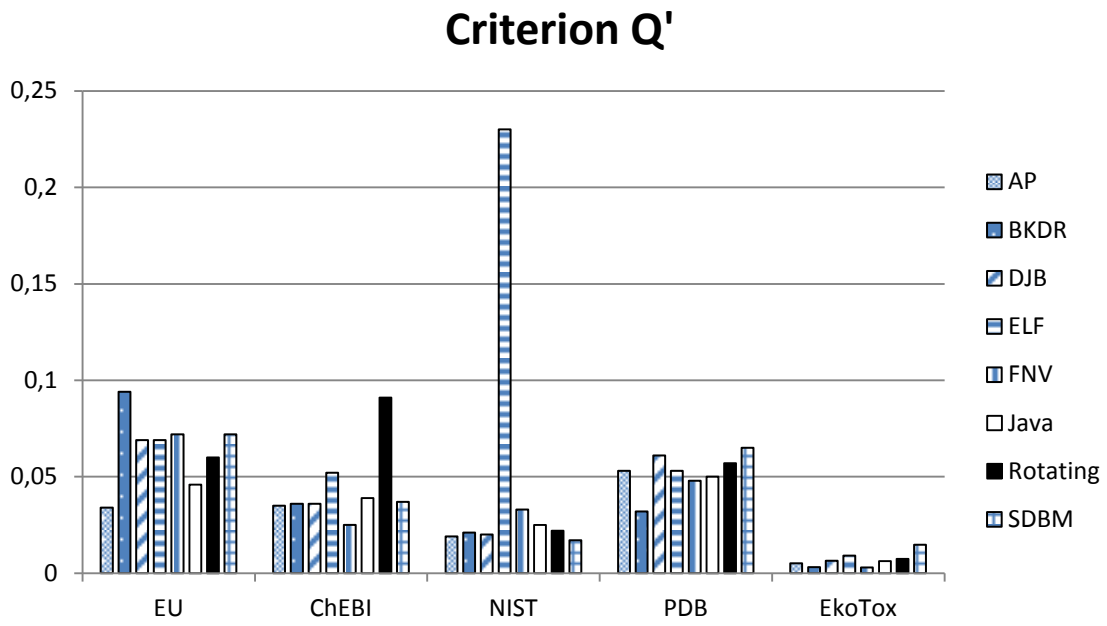

Figure 4: Criterion Q'– comparison against S-Hashing

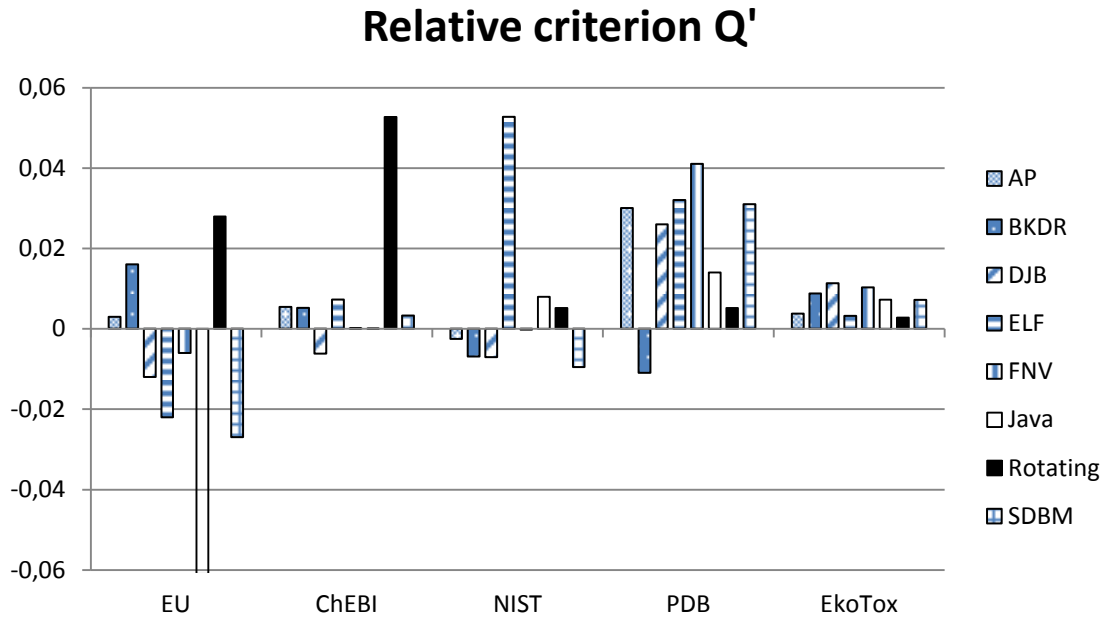

Figure 5: Relative criterion sQ' – comparison against S-Hashing