Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 41 (2011) 5914-5918

ISSN: 2229-712X

Optimization of bloom filter using simulated annealing for spam filtering

Arulanand Natarajan[1], Subramanian S[2] and Premalatha K[3]

[1]Anna University of Technology, Coimbatore, TN, India

[2]Sri Krishna College of Engineering and Technology Coimbatore, TN, India.

[3]Bannari Amman Institute of Technology, Erode, TN, India.

ARTICLE INFO

Article history:

Received: 21 September 2011;

Received in revised form:

29 November 2011;

Accepted: 17 December 2011;

Keywords

Bin Bloom Filter,

Bloom Filter,

Spam word,

Hash function,

Simulated Annealing,

False positive rate.

ABSTRACT

Bloom Filter (BF) is a simple but powerful data structure that can check membership to a static set. The trade-off to use Bloom filter is a certain configurable risk of false positives. The odds of a false positive can be made very low if the hash bitmap is sufficiently large. Spam is an irrelevant or inappropriate message sent on the internet to a large number of newsgroups or users. A spam word is a list of well-known words that often appear in spam mails. The proposed system of Bin Bloom Filter (BBF) groups the words into number of bins with different false positive rates based on the weights of the spam words for spam filtering. Simulated Annealing (SA) is stimulated by an analogy to annealing in solids. It is used to search for feasible solutions and converge to an optimal solution. In this paper SA is applied to minimize the total membership invalidation cost of BBF. The experimental results are analyzed for various sizes of bins. The results show that, the BBF using SA with different false positive rate has lower total membership invalidation cost than the standard BF.

Introduction

A spam filter is a program that is used to detect unsolicited and unwanted email and prevent those messages from getting into user's inbox. A spam filter looks for certain criteria on which it stands decisions. For example, it can be set to look for particular words in the subject line of messages and to exclude these from the user's inbox. This method is not effective, because often it is omitting perfectly legitimate messages and letting actual spam through. The strategies used to block spam are diverse and includes many promising techniques. Some of the strategies like black list filter, white list / verification filters rule based ranking and naïve bayesian filtering are used to identify the spam.

A Bloom filter presents a very attractive option for string matching (Bloom 1970). It is a space efficient randomized data structure that stores a set of signatures efficiently by computing multiple hash functions on each member of the set. It queries a database of strings to verify for the membership of a particular string. The answer to this query can be a false positive but never be a false negative. The computation time required for performing the query is independent of the number of signatures in the database and the amount of memory required by a Bloom filter for each signature is independent of its length (Feng et al 2002).

This paper presents a BBF which allocates different false positive rates to different strings depending on the significance of spam words and gives a solution to make the total membership invalidation cost minimum. BBF groups strings into different bins via smoothing by bin means technique. The number of strings to be grouped and false positive rate of each bin is identified through SA which minimizes the total membership invalidation cost. This paper examines different number of bins for given set of strings, their false positive rates and number of strings in every bin to minimize the total membership invalidation cost.

The organization of this paper is as follows. Section 2 deals with the standard BF. Section 3 presents the SA technique. Section 4 explains the optimized BBF using SA. Performance evaluation of the BBF with standard BF is discussed in section 5.

Bloom Filter

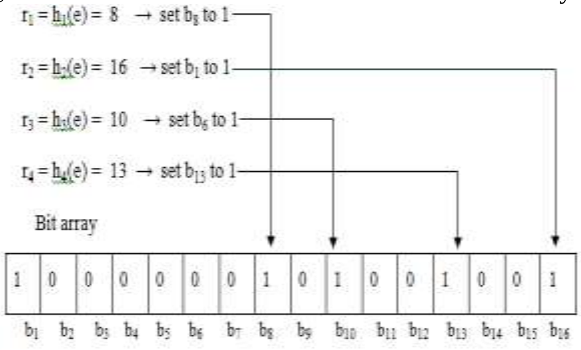
Bloom filters (Bloom 1970) are compact data structures for probabilistic representation of a set in order to support membership queries. This compact representation is the payoff for allowing a small rate of false positives in membership queries which might incorrectly recognize an element as member of the set.

Given a string S the BF computes k hash functions on it producing k hash values and sets k bits in an m-bit long vector at the addresses corresponding to the k hash values. The value of k ranges from 1 to m. The same procedure is repeated for all the members of the set. This process is called programming of the filter. The query process is similar to programming, where a string whose membership is to be verified is input to the filter. The bits in the m-bit long vector at the locations corresponding to the k hash values are looked up. If at least one of these k bits is not found in the set then the string is declared to be a nonmember of the set. If all the bits are found to be set then the string is said to belong to the set with a certain probability. This uncertainty in the membership comes from the fact that those k bits in the m-bit vector can be set by any other n-1 members. Thus finding a bit set does not necessarily imply that it was set by the particular string being queried. However, finding a bit not set certainly implies that the string does not belong to the set.

In order to store a given element into the bit array, each hash function must be applied to it and, based on the return

Tele:
E-mail addresses: arulnat@yahoo.com, dsraju49@gmail.com, kpl_barath@yahoo.co.in

value r of each function (r_1, r_2, … , r_k), the bit with the offset r is set to 1. Since there are k hash functions, up to k bits in the bit array are set to 1 (it might be less because several hash functions might return the same value). Figure 1 is an example where m=16, k=4 and e is the element to be stored in the bit array.



$r_1 = h_1(e) = 8 \rightarrow$ set b_8 to 1
$r_2 = h_2(e) = 16 \rightarrow$ set b_1 to 1
$r_3 = h_3(e) = 10 \rightarrow$ set b_6 to 1
$r_4 = h_4(e) = 13 \rightarrow$ set b_{13} to 1
Bit array
1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1
b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15} b_{16}

Figure 1 Bloom Filter

One important feature of BF is that there is a clear tradeoff between the size of the filter and the rate of false positives. The false positive rate of BF is

$$f = \left(1- e^{-kn/m}\right)^k = \exp(k\ln(1-e^{-kn/m})) \qquad (1)$$

Let $g = k\ln(1-e^{-kn/m})$. Minimizing the false positive probability f is equivalent to minimizing with respect to k.

$$\frac{dg}{dk} = \ln\left(1-e^{-\frac{kn}{m}}\right) + \frac{kn}{m}\frac{e^{-kn/m}}{1-e^{-kn/m}} \qquad (2)$$

The derivative equals 0 when k_{min}=(1n2)(m/n). In this case the false positive probability f is:

$$f(k_{min}) = (1-p)^{k_{min}} = \left(\frac{1}{2}\right)^{k_{min}} = (0.6185)^{m/n} \qquad (3)$$

of course k should be an integer, so k is $\lceil \ln 2 \,.\, (m/n) \rceil$

The BF has been widely used in many database applications (Mullin 1990 ; Mackert and Lohman, 1986). It is applied in networking literature (Broder and Mitzenmacher, 2005).

A BF can be used as a summarizing technique to aid global collaboration in peer-to-peer networks (Kubiatowicz et al., 2000 ; Li et al, 2002 ; Cuena-Acuna et al, 2003). It supports probabilistic algorithms for routing and locating resources (Rhea and Kubiatowicz 2004; Hodes et al, 2002; Reynolds and Vahdat, 2003; Bauer et al, 2004) and share Web cache information (Fan et al,2000). BFs have great potential for representing a set in main memory (Peter and Panagiotis, 2004) in stand-alone applications.

BFs have been used to provide a probabilistic approach for explicit state model checking of finite-state transition systems (Peter and Panagiotis, 2004). It is used to summarize the contents of stream data in memory (Jin et al,2004; Deng and Rafiei,2006), to store the states of flows in the on-chip memory at networking devices (Bonomi et al,2006), and to store the statistical values of tokens to speed up the statistical-based Bayesian filters (Li and Zhong,2006). The variations of BFs are compressed Bloom filters (Mitzenmacher,2002), counting Bloom filters (Fan et al,2000), distance-sensitive Bloom filters (Kirsch and Mitzenmacher,2006), Bloom filters with two hash functions (Kirsch and Mitzenmacher,2006), spacecode Bloom filters (Kumar et al,2004), spectral Bloom filters (Cohen and Matias,2003), generalized Bloom filters (Laufer et al,2005), Bloomier filters (Chazelle et al,2004), and Bloom filters based on partitioned hashing (Hao et al,2007).

Simulated annealing

In an optimization problem, often the solution space has many local minima. A simple local search algorithm proceeds by choosing random initial solution and generating a neighbor from that solution. If it is a minimum fitness transition then the neighboring solution is accepted. Such an algorithm has the drawback of often converging to a local minimum. The SA (Dowsland 1995) avoids getting trapped in a local minimum by accepting cost increasing neighbors with some probability. It solves this problem by allowing worse moves (lesser quality) to be taken some of the time. That is, it allows some uphill steps so that it can escape from local minima. In SA, first an initial solution is randomly generated, and a neighbor is found and is accepted with a probability of min (1, exp (-ΔE/T)), where ΔE is the cost difference and T is the control parameter corresponding to the temperature of the physical analogy and will be called temperature On slow reduction of temperature, the algorithm converges to the global minimum. Among its advantages are the relative ease of implementation and the ability to provide reasonably good solutions for many combinatorial problems. Simulated Annealing is inherently sequential and hence very slow for problems with large search spaces.

Algorithm

```
Set X a initial configuration
Set E as Eval(X)
Set T as high temperature and frozen is false
while (!frozen)
repeat
Choose a random move i from the move set
Set Ei as Eval(move(X, i))
if E < Ei then
set X as move(X, i)
set E as Ei
else accept the move with probability exp(-(ΔE/T) even though things get worse
until the s stem is in thermal equilibrium at T
if ((E is stil decreasing over the last few temperatures)
reduce T
else
assign frozen is true
```

Bloom Filter Optimization using SA

Bin bloom filter

A BBF is a date structure considering weight for spam word. It groups spam words into different bins depending on their weight. It incorporates the information on the spam word weights and the membership likelihood of the spam words into its optimal design. In BBF a high cost bin lower false positive probability and a low cost bin has higher false positive probability. The false positive rate and number of strings to be stored is identified through optimization technique SA which minimize the total membership invalidation cost. Figure 3 shows Bin Bloom filter with its tuple <n,f,w> configuration.

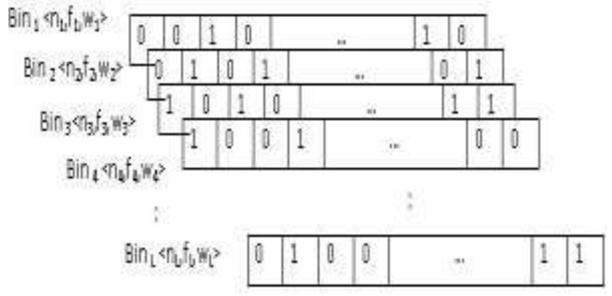


Figure 3 Bin Bloom Filter

Problem Definition

Consider a standard supervised learning problem with a set of training data D = {$<Y_1,Z_1>$,..., $<Y_i, Z_i>$, … ,$< Y_r ,Z_r >$} , where Y_i is an instance represented as a single feature vector, Z_i = $C(Y_i)$ is the target value of Y_i , where C is the target function. Where Y_1, Y2,… , Y_r set of text document collection C is a class label to classify into spam or legitimate (non-spam). The collection is represented into feature vector by the text documents are converted to normalized case, and tokenized them, splitting on non-letters. The stop words are eliminated. The spam weights for words are calculated from the set. This weight value indicates its probable belongings to spam or legitimate. The weight values are discretized and assigned for different Bins. The tuple to describe the Bin Bloom Filter is, $\{\{n_1, n_2, …, n_L\}, \{w_1, w_2,…, w_L\}, m, \{k_1, k_2, …, k_L\}, \{f_1, f_2, …, f_L\}\}$. It is an optimization problem to find the value of n and f that to minimize the total membership invalidation cost. For membership testing the total cost of the set is the sum of the invalidation cost of each subset. The total membership invalidation cost (Xie et al., 2005) is given as,

$E= n_1f_1w_1 + n_2f_2w_2 +……+ n_Lf_Lw_L$

The total membership invalidation cost or the Evaluation function is

$$E(L) = \sum_{i=1}^{L} n_i w_i f_i \quad (4)$$

.where $\sum_{i=1}^{L} n_i = N$

N- Total number of strings in a spam set.

$$f_i = \left(\frac{1}{2}\right)^{\ln 2\left(m \big/ \sum_{j=1}^{i} n_j\right)}$$

$$r_i = \ln(f_i) \quad (1 \le i \le L)$$

Weight assignment

The first step for assigning weight to spam words is estimating the word probability that depends on word frequency. Word frequency is measured by the number of occurrences of a specific word in the document. Estimating probabilities is achieved using Bayes conditional probability theorem according to which the probability of a word given that the message is spam can be estimated as follows:

$$P_s = \frac{\frac{f_s}{N_s}}{\frac{f_{ns}}{N_{ns}} + \frac{f_s}{N_{ns}}} \quad (5)$$

$$P_{ns} = \frac{\frac{f_{ns}}{N_{ns}}}{\frac{f_{ns}}{N_{ns}} + \frac{f_s}{N_{ns}}} \quad (6)$$

P_s is the probability of a word given the mail is spam.
P_{ns} is the probability of a word given the mail is legitimate.
f_s is the frequency of word in the spam documents.
f_{ns} is frequency of words in the legitimate documents.
N_s is the total spam documents.
N_{ns} is the total legitimate documents.

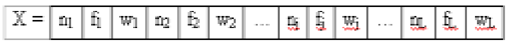
The next step is calculating word weights. Estimating a weight for each word is based on its frequency and its probability in spam mail documents and non-spam mail documents. The weight of every word is estimated using the formula:

$$\text{weight}_{\text{word}} = \frac{P_s}{P_{ns}} \quad (7)$$

This weight value is based on text collection containing spam messages and non-spam messages. The word weights are estimated for spam list during the training process and stored in a separate text document.

Initial solution

In the context of Bin bloom filter, a chromosome represents number of bloom filters with its number of words to be stored, false positive rate and its weight. That is, each chromosome X_i, is constructed as follows:



X =	n_1	f_1	w_1	n_2	f_2	w_2	…	n_j	f_j	w_j	…	n_L	f_L	w_L

Figure 4 Initial Solution

where n_j f_j and w_j refer respectively the number of words, false positive rate and average weight of the jth bin. A set of 3 genes <n,f,w> encodes a protein – a trait, that is a single bin. The false positive rate f_j can be obtained from equation (6) where n_j is drawn from the particle dimension, m is known in advance and k is calculated from equation (8).

SA solves problem by allowing worse moves to be taken some of the time. That is, it allows some uphill steps so that it can escape from local minima.

The maximum cost function difference between one neighbour and another may be taken as a starting temperature. Another method, suggested in (Rayward-Smith, 1996), is to start with a very high temperature and cool it rapidly until about 60% of worst solutions are being accepted. This forms the real starting temperature and it can now be cooled more slowly. Dowsland (1995) suggested for heating the system until a certain proportion of worse solutions are accepted and then slow cooling can start.

The decrement of temperature is important to the success of SA. One way to decrement the temperature is a simple linear method. An alternative is a geometric decrement is $t = t\alpha$ where $\alpha < 1$.

Experimental results

The total number of strings taken for testing is 3000 and their weights are ranging from 0.0005 to 5. The size of the BF is 1024. These experimental values are tested for bin size 10,11 and 12. The Temperature T is assigned to BF size 1024 and T is decremented using the formula $t = t\alpha$ where α=0.9.

Since Bloom Filter allows false positive, the membership invalidation cost is unavoidable. For BBF, the total membership invalidation cost is expressed in equation (4). In standard BF, different weights in different bins into consideration, the total membership invalidation cost is then as follows.

$F_{standard}= (n_1w_1 + n_2w_2 +……+ n_Lw_L)f$

$$F_{standard}(L) = f\sum_{i=1}^{L} n_i w_i$$

Figures 5,6 and 7 show the membership invalidation cost for standard BF with BBF using SA for the bin sizes 10,11 and 12 respectively.

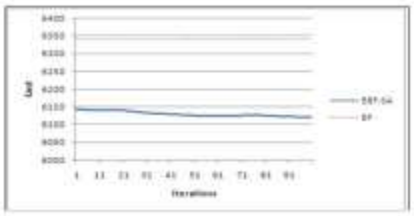


Figure 5 Membership invalidation cost for bin size 10

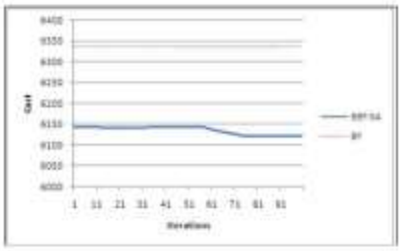


Figure 6 Membership invalidation cost for bin size 11

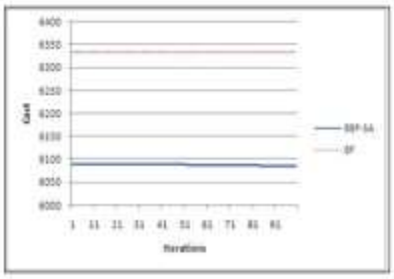


Figure 7 Membership invalidation cost for bin size 10

Table 1 shows the values obtained at 100th iteration of the proposed work. The first column represents the bin size; the second column, number of strings in each bin; the third column, average weight of the strings present in a bin; the fourth column, false positive rate of each bin. The fifth and sixth column show the total membership invalidation cost of each bin in BBF and standard BF respectively. The experimental results exhibit that the total membership invalidation cost of the BBF remarkably less value compared with the original standard BF.

Conclusion

Bloom filters are simple randomized data structures that are useful in practice. The BBF is an extension of BF, and inherits the best feature of BF such as time and space saving. The BBF treats strings in a set in a different way depending on their significance, groups the strings into bins and allocates different false positive rate to different bins. Important spam words have lower false positive rate than less significant words. SA is an optimization technique which chooses a random move from the neighbourhood. If the move is better than its current position then simulated annealing will always take it. If the move is worse then it will be accepted based on some probability. SA technique is applied in this paper to minimize the total membership invalidation cost. Experiment results show that the total membership invalidation cost in spam filtering with different false positive rate of BBF performs better than standard BF.

References

[1] Anayat, S, Ali, A and Ahmad, H.F. Using a probable weight based Bayesian approach for spam filtering , Proceedings of INMIC, 2004, 340-345

[2] Dowsland, K.A., Simulated Annealing. In Modern Heuristic Techniques for Combinatorial Problems (ed. Reeves, C.R.), *McGraw-Hill*, (1995).

[3] Bauer D, Hurley P, Pletka R, and Waldvogel M, Bringing Efficient Advanced Queries to Distributed Hash Tables, Proc. IEEE Conf. Local Computer Networks, 2004, 6-14

[4] Bloom B, Space/time tradeoffs in hash coding with allowable errors, Communications of the ACM, 13, 1970, 422–426.

[5] Bonomi F, Mitzenmacher M, Panigrahy R, Singh S, and Varghese G, Beyond Bloom Filters: From Approximate Membership Checks to Approximate State Machines, Proc. ACM SIGCOMM, 2006 , 315-326.

[6] Broder A and Mitzenmacher M. Network Applications of Bloom Filters: A Survey, Internet Math., 1(4), 2005, 485-509.

[7] Chazelle B, Kilian J, Rubinfeld R, and Tal A, The Bloomier Filter: An Efficient Data Structure for Static Support Lookup Tables, Proc. Fifth Ann. ACM-SIAM Symp. Discrete Algorithms (SODA), 2004, 30-39.

[8] Cohen S and Matias Y, Spectral Bloom Filters, Proc. 22nd ACM SIGMOD, 2003, 241-252.

[9] Cuena-Acuna F.M, Peery C,Martin R.P, and Nguyen T.D, PlantP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities, Proc. 12th IEEE Int'lSymp. High Performance Distributed Computing, 2003, 236-249

[10] Deng F and Rafiei D, "Approximately Detecting Duplicates for Streaming Data Using Stable Bloom Filters," Proc. 25th ACMSIGMOD, 2006, 25-36.

[11] Fan L, Cao P, Almeida J, and Broder A, Summary Cache: A Scalable Wide Area Web Cache Sharing Protocol, IEEE/ACM Trans. Networking, 8(3), 2000, 281-293.

[12] Feng W,.Shin K.G, Kandlur D.D. & D.Saha, "The BLUE active queue management algorithms", IEEE/ACM Transactions on Networking, 10, 2002, 513 – 528.

[13] Hao F, Kodialam M, and Lakshman T.V, Building High Accuracy Bloom Filters Using Partitioned Hashing, Proc. SIGMETRICS/Performance, 2007, 277-287 . Hodes T.D, Czerwinski S.E, and Zhao B.Y, An Architecture for Secure Wide Area Service Discovery, Wireless Networks, vol. 8, nos. 2/3, 2002, 213-230.

[14] Jin C, Qian W, and Zhou A, Analysis and Management of Streaming Data: A Survey, J. Software, 15(8), 2004, 1172-1181.

[15] Kirsch A and Mitzenmacher M, Building a Better Bloom Filter, Technical Report tr-02-05.pdf, Dept. of Computer Science, Harvard Univ,2006.

[16] Kirsch A and Mitzenmacher M, Distance-Sensitive Bloom Filters, Proc. Eighth Workshop Algorithm Eng. and Experiments (ALENEX '06), 2006.

[17] Kumar A, Xu J, Wang J, Spatschek O, and Li L, Space-Code Bloom Filter for Efficient Per-Flow Traffic Measurement, Proc. 23rd IEEE INFOCOM, 2004, 1762-1773.

[18] Kubiatowicz J Bindel D, Chen, Y Czerwinski S, Eaton P, and Geels D, Oceanstore: An Architecture for Global-Scale Persistent Storage," ACM SIGPLAN Notices, 35(11), 2000, 190-201.

[19] Li J, Taylor J, Serban L, and Seltzer M, Self-Organization in Peer-to-Peer System, Proc. ACM SIGOPS, 2002.

[20] Li K and Zhong Z, Fast Statistical Spam Filter by Approximate Classifications, Proc. Joint Int'l Conf. Measurement and Modeling of Computer Systems, SIGMETRICS/Performance, 2006, 347-358.

[21] Laufer R.P, Velloso P.B, and Duarte O.C.M.B, GeneralizedBloom Filters, Technical Report Research Report GTA-05-43, Univ. of California, Los Angeles (UCLA), 2005.

[22] Mackert L.F. and Lohman G.M., Optimizer Validation and Performance Evaluation for Distributed Queries, Proc. 12th Int'l Conf. Very Large Data Bases (VLDB), 1986, 149-159.

[23] Mitzenmacher M, Compressed Bloom Filters, IEEE/ACM Trans.Networking, 10(5) 2002, 604-612.

[24] Mullin J.K, Optimal Semijoins for Distributed Database Systems, IEEE Trans. Software Eng., 16, 1990, 558-560.

[25] Peter C.D and Panagiotis M, Bloom Filters in Probabilistic Verification, Proc. Fifth Int’l Conf. Formal Methods in Computer- Aided Design, 2004, 367-381.
[26] Reynolds P and Vahdat A, Efficient Peer-to-Peer Keyword Searching, Proc. ACM Int’l Middleware Conf., 2003, 21-40.
[27] Rhea S.C and Kubiatowicz J, Probabilistic Location and Routing, Proc. IEEE INFOCOM, 2004, 1248-1257.
[28] Xie K., Min Y., Zhang D., Wen J., Xie G. & Wen J, Basket Bloom Filters for Membership Queries, Proceedings of IEEE Tencon’05,2005, 1-6.

Table 1: Values obtained from SA at 100th iteration

Bin size	Number of Strings	Average String weight	False positive rate	Cost of BBF	Cost of Standard BF
10	96, 129 174, 212 319, 321 383, 385 490, 491	4.906, 4.663 4.424, 4.101 3.694, 3.190 2.607, 1.966 1.250, 0.385	0.0059, 0.0221, 0.0592, 0.0982 0.2139, 0.2159, 0.2767, 0.2786 0.3664, 0.3671	6119.99	6340.83
11	45, 45 124, 169 258, 259 356, 357 437, 438 512	4.956, 4.757 4.677, 4.445 4.095, 3.691 3.211, 2.619 1.961, 1.242 0.402	1.79E-05, 1.79E-05 0.0189, 0.0544 0.1485, 0.1496 0.2511, 0.2521 0.3244, 0.3252 0.3825	6122.02	6336.61
12	91, 91 91, 146 179, 220 276, 276 370, 371 444, 445	4.910, 4.688 4.535, 4.364 4.090, 3.789 3.404, 2.952 2.417, 1.810 1.127, 0.345	0.0045, 0.0045 0.0045, 0.0344 0.0640, 0.1069 0.1682, 0.1682 0.2646, 0.2655 0.3302, 0.3310	6085.45	6334.59