# Fast Estimation of Fractal Dimension and Correlation Integral on Stream Data

Angeline Wong
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
alwong@andrew.cmu.edu

Leejay Wu
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
lw2j@cs.cmu.edu

Phillip B. Gibbons
Intel Research Pittsburgh
417 South Craig Street, Suite 300
Pittsburgh, PA 15213
phillip.b.gibbons@intel.com

## ABSTRACT

Given a cloud of $N$ points in an $E$-dimensional space, we often need to estimate the intrinsic dimensionality $D$ of this cloud. For example, a set of points in 3-dimensional space all following along a straight line has intrinsic (or *fractal*) dimensionality $D$=1. Non-integer fractal dimensionality appears pervasively in nature.

In this paper we give a very fast method to estimate the fractal dimensionality of the points in a data stream. Algorithms to estimate the fractal dimension exist, from the straightforward quadratic algorithm, to the faster O($N$log$N$) or even O($N$) algorithms that use the so-called box-counting method. However, these algorithms require $\Omega$(N) space, and hence are ill-suited to semi-infinite streams of data. In this paper we propose an algorithm, based on a "tug-of-war" idea, which computes the fractal dimension in a single pass over the dataset using only constant memory. Experimental results on synthetic and real world data sets demonstrate the effectiveness of our algorithm.

## Categories and Subject Descriptors

G.1.2 [**Numerical Analysis**]: Approximation – *wavelets and fractals, linear approximation, special function approximations*; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation – *online computation, probabilistic computation*; E.2 [**Data Storage Representations**]: *hash-table representations*; E.1 [**Data Structures**]: *arrays*; G.3 [**Probability and Statistics**]: *correlation and regression analysis*; I.5.5 [**Pattern Recognition**]: Implementation – *interactive systems*.

## General Terms

Algorithms, Theory, Design, Measurement, Performance, Reliability, Verification, Experimentation.

## Keywords

Approximation algorithms, Databases, Randomized algorithms, Fractal dimension, Box counting.

## 1. INTRODUCTION

The *fractal dimension* of a set of points is an important measure of the intrinsic dimensionality of the points. It has been shown to facilitate selectivity estimation, range queries [7,15], nearest-neighbor queries [3,4,13], and similarity searches [3], and can be used in dimensionality reduction [17] and outlier detection [14]. It also assists the characterization of workloads [12] such as disk requests [18], processor utilization, or network traffic [6,9,18]; and the modeling of random walks, such as stock indexes over time. The fractal dimension also aids in shape classification problems, such as preliminarily diagnosing cancerous masses [5,10].

The development of fast approximate algorithms to calculate fractal dimensions has reduced the computational complexity of estimating fractal dimensions from O($N^2$), where $N$ is the number of points in the dataset, to linear O($N$) [17]. However, the memory costs have remained at $\theta$($N$). This makes these previous algorithms ill-suited for semi-infinite streams of data, which have been attracting increasing interest thanks to network monitoring and sensor settings [2,8]. In this paper, we propose an algorithm, Tug-of-War, which combines O($N$) time and O(*1*) space costs, and we provide an experimental study of its effectiveness in estimating fractal dimensions. In the next section, we describe in greater detail fractals, fractal dimensionality and current algorithms for computing that measure. Following the background comes our description of Tug-of-War, and then our experimental results.

## 2. BACKGROUND AND DEFINITION OF FRACTALS AND FRACTAL DIMENSIONS

Suppose we have a self-similar dataset of $N$ points $a_i$ in $\Re^E$, such as the Sierpinski Triangle (Figure 1(a)). What is its intrinsic dimensionality? While with ordinary Euclidian point sets such as uniform lines, planes, and solids, the intrinsic dimensionality would be the corresponding obvious integer (e.g., $D$=1 for Figure 1(b)), for real point sets a fractional dimension – or fractal dimension – may better quantify the intrinsic dimensionality. Non-integer fractal dimensionality appears pervasively in nature: the periphery of clouds and rain patches ($D$=1.3), coast-lines ($D$=1.1 to 1.58 for Norway), the surface of mammalian brains ($D$=2.6-2.7), river basins, the bark of trees, stock prices and random walks ($D$=1.5), the human respiratory system ($D$=2.9), the cardiovascular system ($D$=3) [11,16], the boundary of malignant and benign tumors [5], and much, much more. The "correlation"

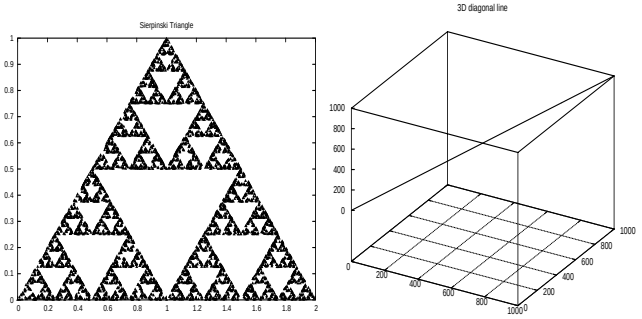


**Figure 1. Theoretical fractals: (a) 5000 points of the Sierpinski Triangle, (b) a line in 3D spage**

fractal dimension $D_2$ for a point-set that shows self-similarity in the range of scales ($r_1$, $r_2$) is defined as follows:

$$D_2 \equiv \frac{\partial \log \sum_i p_i^2}{\partial \log(r)} \qquad r \in (r_1, r_2)$$

where $p_i$ is the occupancy of, or number of points contained in, the $i$-th cell when the $E$-dimensional address space is divided into (hyper)-cubic grid cells of side $r$ [3]. The method we propose approximates this correlation fractal dimension $D_2$.

## 2.1 Survey of Methods for Estimating Fractal Dimensions

The following subsections give a brief explanation of two salient methods for computing $D_2$ fractal dimensions. Pair-counting provides an exact way to calculate fractal dimensions, but needs to enumerate all O($N^2$) pairs, driving research to find faster algorithms. Box-counting incurs only O($N$) computational cost, but sacrifices accuracy.

### *2.1.1 Pair-counting*

Pair-counting calculates how many pairs are within a given distance of each other. The average number of neighbors within a given radius $r$, $C_r$, is exactly the total number of pairs within distance $r$ of each other, divided by the number of points $N$ in the set:

$$\textit{average \# neighbors} (\leq r) = (\textit{total \# pairs} (\leq r))/N$$

Self-similar sets obey a power-law relationship

$$C_r \text{ proportional to } r^{D_2} \qquad [3]$$

within limits – below or above certain radii, a finite set must necessarily stray from strict adherence to that law. Therefore, the fractal dimension can be computed as the slope of the linear relationship between log($C_r$) and log($r$). We can also compute the correlation between those two quantities within the region of radii where the set is in fact linear or nearly so; this correlation then serves as a measure of reliability and self-similarity.

Computing the quantity $D_2$ exactly requires determining the distance between every two points. While the distances need not be stored (we may keep counters for each and every distance $r$ one intends to use), they must be computed at least once to produce an exact answer. This O($N^2$) computational cost proves prohibitive for realistically large data sets. Consequently, one compromises on accuracy to gain performance.

### *2.1.2 Box-counting*

The practical box-counting algorithm provides one such compromise [3,16,17]. This algorithm derives its name from the imposition of nested hypercube grids over the data, followed by counting the occupancy of each grid cell [3], thus focusing on individual points instead of on pairs. The sum of squared occupancies $S_2(r)$ for a particular grid side length $r$ is defined as: $S_2(r) \equiv \sum_i C_i^2$ , where $C_i$ is the count of points in the $i$th cell. Substituting these counts for the pair counts in the same power-law relationship estimates the fractal dimension. Given sufficient space to store all the counters simultaneously, all counts can be computed in a single pass over the data. Even if performing one pass per radius to minimize storage cost, the computational cost is still only linear with respect to $N$ [17]. However, even this algorithm still has a storage complexity of $\theta(N)$, as every datum might end up in its own cell given a sufficiently small radius.

# 3. PROPOSED METHOD: TUG-OF-WAR

Accepting an O($N$) computational cost seems reasonable, as one must consider every point at least once unless using sampling or similar approaches. However, we can drastically reduce the storage cost to the point where it proves *constant* with respect to $N$. Our proposed method provides such space-efficient performance without sacrificing an O($N$) runtime performance, returning reasonably accurate results while permitting computation to occur within the confines of a single pass over the data.

## 3.1 Preliminaries

Given a set of grid-cell identifiers, we want to estimate the sum of squared occupancies $S_2(r)$ for a range of grid sides $r$ and do so in a single pass through the data while using a small, fixed amount of memory. Theorem 2.2 of [1] describes a space-efficient, randomized algorithm to estimate the second moment, $F_2$ (which is $S_2(r)$ for a given grid side $r$), given a sequence $A = (a_1,\ldots, a_N)$ of members of a set $M$, in a single pass, performing a constant number of arithmetic and finite field operations. The algorithm generates a set of random, four-wise independent binary hash functions $h_{ij}$ that are used to compute a value $Z$, the square of which ($Z^2$) approximates $F_2$. $Z^2$ is the median of $s_2$ random variables $Y_1$, $Y_2$, …, $Y_{s2}$. Each $Y_i$ is the average of $s_1$ random variables $X_{ij}$ : $1 \leq j \leq s_1$, where the $X_{ij}$ are derived as follows: In one pass over the data, for each data value $v$, add 1 to $X_{ij}$ if $h_{ij}(v) = 1$ and otherwise subtract 1 from $X_{ij}$ [1]. This technique is also referred to as taking a *random sketch* of the data [8].

The theorem also provides probably approximately correct (PAC) bounds for computing $F_2$ using the specified algorithm. For any positive integral values of $s_1$ and $s_2$, let $\lambda = \sqrt{(16/s_1)}$ and $\varepsilon = 1/(e^{(s_2/2)})$. Then, the probability that the estimated moment deviates from the true moment $F_2$ by more than ($\lambda * F_2$) is at most $\varepsilon$ [1]. When $s_1 = 30$ and $s_2 = 5$, $\lambda = 0.73$ and $\varepsilon = 0.082$.

## 3.2 Proposed Algorithm

Our Tug-of-War algorithm efficiently approximates the pair counts by extending this algorithm to compute $F_2$ for multi-dimensional points and for a *range* of grid sides *r*. Let $s_1$ be an integer positively correlated with how much accuracy we require, while $s_2$ similarly influences confidence.

Now suppose we have a self-similar dataset *M* of *N* points $a_i$ in $\Re^E$, from which we can form a sequence $A = (a_1, a_2, \ldots, a_N)$. Since each point $a_i$ is *E*-dimensional, we can expand each *ai* to form a sequence $A' = (a_{11}, \ldots, a_{1E}, a_{21}, \ldots, a_{NE})$ of members of *M*, with each member expressed as a sequence of its coordinates. The members of *M* will still be processed in a single pass. Then, for each radius, we generate $s_1 s_2$ random four-wise independent hash functions of the form

$$result = \sum_j a^j x_{ij}^3 + b^j x_{ij}^2 + c^j x_{ij} + d^j \mod q$$

where the coefficients *a,b,c,d* are random primes, ensuring that the polynomial is irreducible [1], and the inputs will be $x_{ij} = \lfloor a_{ij}/r \rfloor$ with $j$ from 1 to *E*. For each radius, we maintain one counter for each of the hash functions, which will give us $|R| = |\{r\}|$ groups of $s_1 s_2$ random variables from which to calculate $|R|$ values of $Z^2$. After each *result* is calculated, it is mapped to –1 or 1 according to its parity. This value is then added to the counter corresponding to the hash function and radius used. The final estimate for the second moment for each radius is $Z^2$, the median of $s_2$ values, each of which is the mean of $s_1$ squares of the values held in the counters. Since $F_2 = S_2(r)$, these estimates give the sum of squared occupancies from which we can compute the fractal dimension.

## 3.3 Computational and Memory Complexities

The computational complexity of the Tug-of-War method is linear in the size *N* of the database or current data stream. The runtime is dependent on *N*, the embedding dimensionality of the *E*, the number of radii $|R| = |\{r\}|$, the accuracy parameter $s_1$, and the confidence parameter $s_2$. *E*, $|R|$, $s_1$, and $s_2$ are independent of *N*, so the runtime complexity of the Tug-of-War method is $O(E*N*r*s_1*s_2)$, or O(*N*) with respect to *N*.

The Tug-of-War method utilizes an amount of memory constant with respect to *N*. This amount is determined by the parameters $|R|$, $s_1$, and $s_2$, which correspond to the number of radii, accuracy variables, and confidence intervals used, but is independent of the size of the database, *N*.

The implemented four-wise independent hash functions each use four hash keys, and one hash function is used for each of the $s_2*s_1*|R|$ counters. Thus the amount of memory required for the hash keys is $4*s_2*s_1*|R|*$sizeof(int), which is $O(s_2*s_1*|R|)$, or O(1) with respect to *N*. The amount required for the counters is $s_2*s_1*|R|*$sizeof(long), which is also $O(s_2*s_1*|R|)$, so again O(1) with respect to *N*. Additional, constant overhead is needed to compute the hash values. The total memory complexity for the Tug-of-War method is therefore O(1) with respect to *N*, but linear with respect to $|R|$, $s_1$, or $s_2$.

Since all the counters and functions may fit in memory simultaneously, the Tug-of-War method can process all the data in a single pass. Streams should present no problems, since at any point one can economically compute the median of means of the counters for each of the radii. Then, to estimate the appropriate partial derivative, we perform a linear regression on the logarithms of the per-radius estimates. None of this scales with the number of points previously encountered, so every update should have the same reasonable cost.

# 4. RESULTS

In this section, we present empirical verification of the speed and accuracy of the Tug-of-War algorithm. Our C++-based implementation was tested on both real world and synthetic data on the Linux platform, with 31 different radii, an accuracy parameter $s_1$ of 30, and confidence parameter $s_2$ of 5.

## 4.1 Quality of the Estimation

Since the fractal dimensions of Euclidean objects equal their Euclidean dimensions, estimating the fractal dimension of a diagonal line provides a simple first test. The test diagonals consist of 1000 points with *E*-dimensional embedding $(\{x_1, .., x_E\} \mid x \in [1..1000])$. The Tug-of-War method estimated $D_2$ fractal dimensions 1.045, 1.050, and 1.038 with correlations 0.997, 0.999, and 0.993 for *E* = 2, 3, and 4, respectively. Further evidence that the Tug-of-War method performs correctly comes from a comparison of the log-log plots produced by this and the box-counting methods (Figure 2). The estimated fractal closely matches the theoretical value of 1, the reported correlations approach 1, and the critical linear slopes within the log-log plots prove nearly identical.

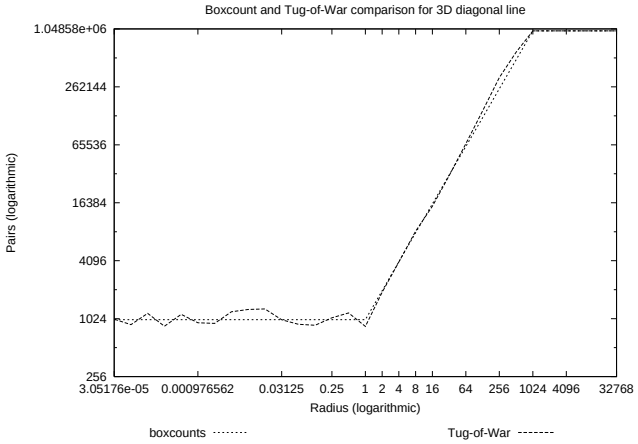


**Figure 2. Comparison of box-count and Tug-of-War log-log plots for diagonal line with E=3**

### *4.1.1 Synthetic Datasets*

Additional testing was performed with more complex fractals to again compare experimental estimates of the fractal dimension and theoretical results. The results of two well-known fractals, the Sierpinski Triangle and the Koch Snowflake (Figure 3), are discussed here.

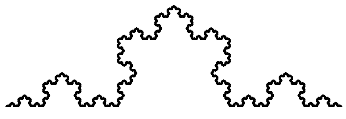

**Figure 3. Koch Snowflake**

The Sierpinski5K set consists of the first 5000

points generated through a breadth-first recursive generation of the Sierpinski triangle. On this set, the Tug-of-War method estimates $D_2$ as 1.520 with a correlation of 0.999, making it reasonably close to the theoretical value for the Triangle, $\log_2 3 \approx 1.585$ [11]. Previously measured $D_2$ for this dataset was 1.587, using the box-counting method.

The theoretical fractal dimension of the Koch Snowflake is $\log_3 4 \approx 1.262$. For a Koch dataset containing 16,385 points, Tug-of-War produces an estimated $D_2$ of 1.255 and a correlation of 0.999. A comparison of the box-counting and Tug-of-War log-log plots (Figure 4(a)) further supports the high quality of the results.

### *4.1.2 Real Datasets*

Real point-sets behave like fractals more than one might expect [3]. Three such datasets, Montgomery County, MD and Long Beach County, CA road intersections and an fMRI (Figure 5), are discussed here.

The Montgomery County and Long Beach County datasets contain 27,282 and 36,548 points in a plane, respectively. The 05886 fMRI dataset has 144,768 points in three dimensions. The Tug-of-War method estimates $D_2$ 1.557, 1.758, and 2.521 with correlations 0.996, 0.999, and 0.998 for these datasets, respectively.

As one might indeed expect due to their non-uniformity, these sets have intrinsic dimensionalities significantly less than their embedding dimensionalities [3]. Even discounting the high correlations and close matches of the box-count and Tug-of-War plots, the Tug-of-War results agree with previous measurements (see Figure 6 and Figure 7). While we know of no previous measurements for the fMRI dataset for comparison, previously measured $D_2$ for Montgomery County and Long Beach County datasets were 1.518 and 1.732, respectively [3], using the box-counting method.

Clearly, Tug-of-War provides excellent performance on both the synthetic and real world datasets, yielding reasonable estimates consistent with measurements by other methods.

## 4.2 Running Time and Memory Consumption

Multiple sets of differing size were generated using the Iterative Function System. Figure 8 shows the linear relationship of running time (as the sole user of an Intel® Xeon™ CPU 2.80 GHz processor) to database size, calculated as the product of the number of points, their embedding dimensionality, and number of radii used. The Tug-of-War method's greatest advantage over other computationally linear algorithms used to estimate fractal dimensions is that it utilizes a small, constant amount of memory relative to the size of the database. To compare across algorithms, we calculated the number of counters necessary for the box-counting and Tug-of-War methods. Figure 9 shows that the number of counters for the Tug-of-War method is constant across all radii and demonstrates that the total number of counters required to attain a 99.5% level of accuracy and precision (as measured by the correlation) when processing large databases is significantly less than that of box-counting.

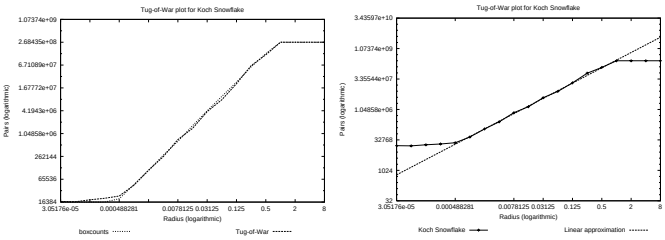


**Figure 4. Tug-of-War log-log plots for the Koch Snowflake: (a) comparison with box-counting, (b) plot with linear approximation**

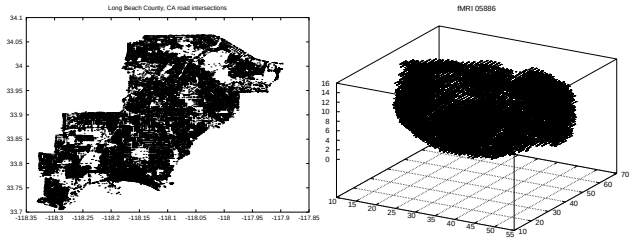


**Figure 5. Real data sets: (a) Long Beach County, CA (E=2), (b) fMRI 05886 (E=3)**

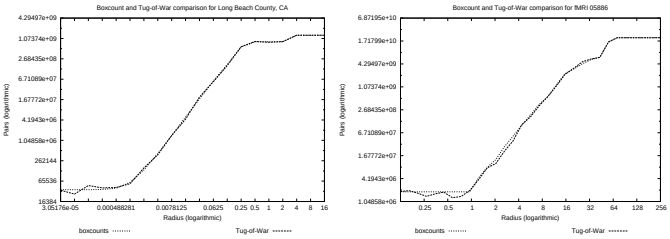


**Figure 6. Comparison of box-count and Tug-of-War plots: (a) Long Beach County, (b) fMRI 05886**

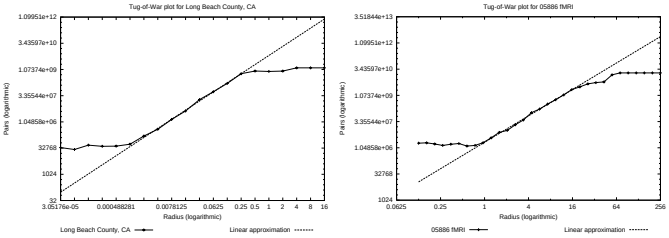


**Figure 7. Tug-of-War plots with linear approximations: (a) Long Beach County, (b) fMRI 05886**

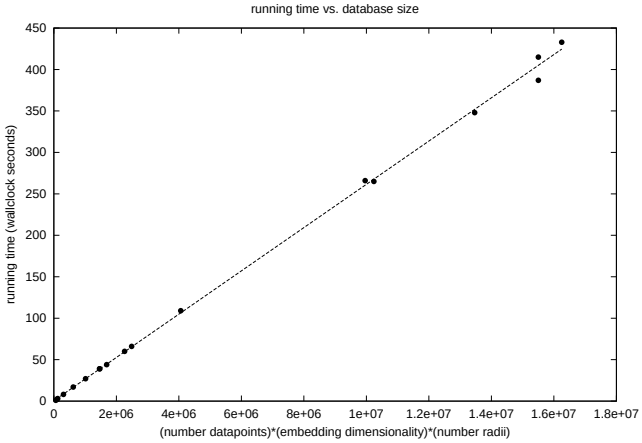


**Figure 8: Running time against database size**

## 5. CONCLUSIONS

The Tug-of-War method is a very fast and space-efficient approximation algorithm for accurately estimating the correlation fractal dimension. Its key features are: it handles multi-dimensional data; it requires but a single pass through the data; it has a computational complexity of O(*N*) and a space complexity of O(1) with respect to *N*; and, it closely matches the accuracy yielded by previous algorithms of similar O(*N*) speed but O(*N*) space complexity. Thus we advocate its use for large data sets and data streams.

## 6. REFERENCES


[1] N. Alon, Y. Matias, M. Szegedy, The Space Complexity of Approximating the Frequency Moments, in: Proc. of 28th ACM Symposium on Theory of Computing (1996) p. 20-29.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, Models and Issues in Data Stream Systems, invited paper in: Proc. of the 2002 ACM Symp. On Principles of Database Systems (PODS 2002), June 2002.

[3] A. Belussi and C. Faloutsos, Estimating the Selectivity of Spatial Queries Using the 'Correlation' Fractal Dimension, in: Proc VLDB, Zurich, Switzerland, September 1995. p. 229-310.

[4] S. Berchtold, et al., Fast Similarity Search in Multimedia Databases, in: SIGMOD Conference, 1997.

[5] C.J. Burdett, et al., Nonlinear Indicators of Malignancy, in Proc. SPIE 1993 – Biomedical Image Processing and Biomedical Visualization, February 1-4, 1993. 1905 (part two of two): p. 853-860.

[6] M. Crovella, A. Bestavros, Self-Similarity in World Wide Web Traffic, Evidence and Possible Causes, in: Sigmetrics, 1996.

[7] C. Faloutsos, I. Kamel, Beyond Uniformity and Independence: Analysis of R-trees Using the Concept of Fractal Dimension, in: Proc. ACM SIGACT-SIGMOD-SIGART PODS, Minneapolis, MN, May 24-26, 1994.

[8] M. N. Garofalakis, J. Gehrke, R. Rastogi, Querying and Mining Data Streams: You Only Get One Look. A Tutorial, SIGMOD Conference 2002: 635.

[9] W. Leland, M. Taqqu, W. Willinger, D. Wilson, On the Self-Similar Nature of Ethernet Traffic (extended version), IEEE/ACM Transactions on Networking, vol. 2., p. 1-15, February 1994.

[10] H. Li, K. Liu, S. Lo, Fractal Modeling and Segmentation for the Enhancement of Microcalcifications in Digital Mammograms, IEEE Transactions on Medical Imaging, vol. 16, pp. 785-798, 1997.


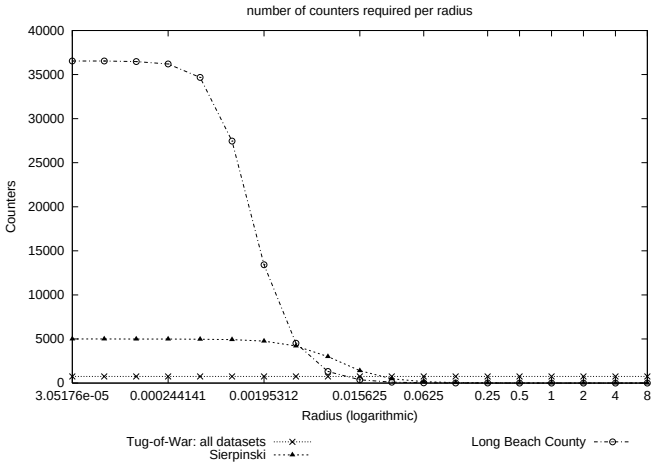


**Figure 9. Counter requirements for box-counting and Tug-of-War methods**


[11] B. Mandelbrot, Fractal Geometry of Nature, W.H. Freeman, New York, 1977.

[12] D. Menascé, B. Abrahão, D. Barbará, V. Almeida, F. Ribeiro, Fractal Characterization of Web Workloads, in: Proc. of Eleventh International World Wide Web Conference, Honolulu, Hawaii, 2002.

[13] B.-U. Pagel, F. Korn, C. Faloutsos, Deflating the Dimensionality Curse Using Multiple Fractal Dimensions, in: ICDE 2000, San Diego, CA.

[14] S. Papadimitriou, H. Kitawaga, P. Gibbons, C. Faloutsos, LOCI: Fast Outlier Detection Using the Local Correlation Integral, in: ICDE 2003, Bangalore, India, March 5-8, 2003.

[15] A. Papadopoulos, Y. Manolopoulos, Performance of Nearest Neighbor Queries in R-trees, in: 6th Int. Conf. On Database Theory (ICDT '97), Delphi, Greece, Jan 8-10, 1997.

[16] M. Schroeder, Fractals, Chaos, Power Laws: Minutes From an Infinite Paradise, W.H. Freeman and Company, New York, 1991.

[17] C. Traina, A. Traina, L. Wu, C. Faloutsos, Fast Feature Selection Using the Fractal Dimension, in: Proc. of XV Brazilian Symposium on Databases (SBBD), Paraiba, Brazil, October 2000.

[18] M. Wang, T. Madhyastha, N. Chang, S. Papadimitriou, C. Faloutsos, Data Mining Meets Performance Evaluation: Fast Algorithms for Modeling Bursty Traffic, ICDE, San Jose, CA, February 2002.