Better Burst Detection

Xin Zhang Dennis Shasha
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
{xinzhang,shasha}@cs.nyu.edu

Abstract

A burst is a large number of events occurring within a certain time window. Many data stream applications require the detection of bursts across a variety of window sizes. For example, stock traders may be interested in bursts having to do with institutional purchases or sales that are spread out over minutes or hours.

In this paper, we present a better framework for elastic burst detection: a family of data structures that generalizes the Shifted Binary Tree, and a heuristic search algorithm to find an efficient structure given the input. We study how different inputs affect the desired structures and the probability to trigger a detailed search. Experiments on both synthetic and real world data show a factor of up to 35 times improvement compared with the Shifted Binary Tree over a wide variety of inputs, depending on the inputs.

1. Introduction

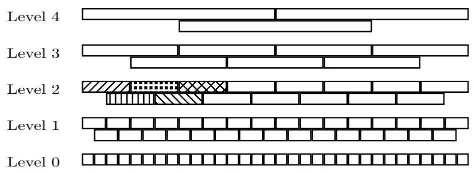
A burst is a large number of events occurring within a certain time window. It's a noteworthy phenomenon in many natural and social processes: stock traders are interested in bursts of trading volume, which are good indicator of the price trend; astrophysicists are interested in gamma ray bursts, which may reflect the occurrence of a supernova. Furthurmore, many data applications require detection of bursts across a variety of window sizes. For example, interesting gamma ray bursts could last several seconds, several minutes or even several days.

The *elastic burst detection* problem [7] is to simutanuously detect bursts across multiple window sizes. When the aggregate in a window exceeds the threshold for this window size, a burst occurs. A naive algorithm is to check each window size of interest one at a time. To detect bursts over k window sizes in a sequence of length N naively requires $O(kN)$ time. In [7], the authors show that a simple data structure called the *Shifted Binary Tree* can beat the naive algorithm by several orders when the probability of bursts is very low.

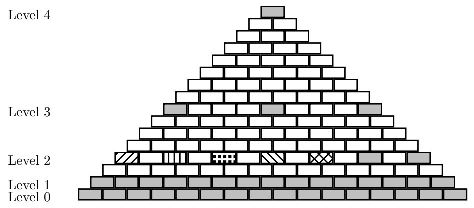
A Shifted Binary Tree includes a binary tree as the base structure just like the Haar wavelet tree. It also includes shifted sublevels to each base sublevel above level 0. The shifted sublevel i is still of length 2^i, but the correspondending window is shifted by 2^{i-1} from the base sublevel. Figure 1.a shows an example.

The overlap between the base sublevels and the shifted sublevels guarantees that all the windows of length w, $w \leq 1 + 2^i$, are included in one of the windows at level $i + 1$. Let $f(w)$ be the threshold for size w. Because the aggregation function is monotonically increasing, if more than $f(2+2^{i-1})$ events are found in a window of size 2^{i+1}, then a detailed search must be performed to check if some subwindow of size w, $2 + 2^{i-1} \leq w \leq 1 + 2^i$, has $f(w)$ events. All bursts are guaranteed to be reported, while many non-burst windows are filtered away without need of a detailed check when the burst probability is very low. However, when bursts are rare but not very rare, the number of fruitless detailed searches grows, suggesting we may want more levels than Shifted Binary Tree provides; conversely, when bursts are exceedingly rare we may need fewer levels. In other words, we want a structure that adapts to the input.

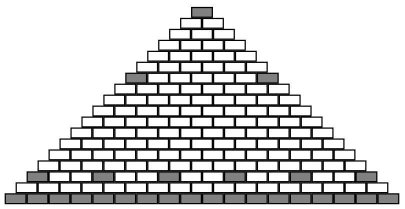
This paper presents a family of multiresolution overlapping data structures, called *Shifted Aggregation Trees*, which generalizes the Shifted Binary Tree and includes many other structures. We then present a heuristic search algorithm to find an efficient data structure given the sample input series and window thresholds. We theoretically analyze and empirically study how different inputs affect the desired structures and the probability to trigger a detailed search. Experiments on both synthetic data and real world data show that the Shifted Aggregation Tree outperforms the Shifted Binary Tree over a variety of inputs, yielding up to a factor of 35 times speedup in some cases. Due to the space limitation, the details missed in this paper can be found in [10].



(a) Shifted Binary Tree



(b) Embed a Shifted Binary Tree (SBT) in an Aggregation Pyramid (AP). Each shaded cell in the AP corresponds to a node in the SBT. The different shadings in level 2 show the one-to-one correspondence.



(c) An example of Shifted Aggregation Tree embedded in the Aggregation Pyramid

Figure 1. Shifted Binary Tree, Aggregation Pyramid and Shifted Aggregation Tree.

2. Shifted Aggregation Tree

2.1. Embed a Shifted Binary Tree in an Aggregation Pyramid

Aggregation Pyramid is an N-level isosceles triangular-shaped data structure built over a time window of size N shown in Figure 1.b and 1.c: level 0 has N cells and is in one-to-one correspondence with the original time series; level 1 has $N-1$ cells, the first cell stores the aggregate of the first two data items (say, data items 1 and 2) in the original time series, the second cell stores the aggregate of the second two data items (data items 2 and 3), etc; and so on. In all, it stores the original time series and all the aggregates for every window size starting at every time point within this time window.

	SBT	SAT
Number of children	2	≥ 2
Levels of children for level $i+1$	i	$\leq i$
Shift at level $i+1$: S_{i+1}	$2 * S_i$	$k * S_i, k \geq 1$
Overlapping window size at level $i+1$: O_{i+1}	window size at level i: w_i	$\geq w_i$

Table 1. Comparing the Shifted Aggregation Tree (SAT) with the Shifted Binary Tree (SBT)

Since in a Shifted Binary Tree, level 0 stores the original time series, and level i stores the aggregates of window size 2^i. Each node in a Shifted Binary Tree has a corresponding cell in the aggregation pyramid. Thus the Shifted Binary Tree can be embedded in the aggregation pyramid as shown in Figure 1.b.

Obviously, there are many other possible subsets of the aggregation pyramid. By using different structures on different data inputs, we can achieve optimal performance by trading off structure maintenance against filtering selectivity.

2.2. Shifted Aggregation Tree Generalizes Shifted Binary Tree

Like a Shifted Binary Tree, a *Shifted Aggregation Tree (SAT)* is a hierarchical tree structure defined on a subset of the cells of an aggregation pyramid. It has several levels, each of which contains several nodes. The nodes at level 0 are in one-to-one correspondence with the original time series. Any node at level i is computed by aggregating some nodes below level i. Two consecutive nodes at the same level overlap in time.

A Shifted Aggregation Tree is different from a Shifted Binary Tree in two ways:

- The parent-child structure defining the topological relationship between a node and its children, i.e. how many children it has and their placements.
- The shifting pattern defining how many time points apart (called the *shift*) two neighboring nodes at the same level are.

Table 2.2 gives a side-by-side comparison of the difference between a SAT and a SBT. Clearly, a SBT is a special case of a SAT. Figure 1.c shows one example of Shifted Aggregation Trees.

A Shifted Aggregation Tree has a similar property to a Shifted Binary Tree: *any window of size* w, $w \leq h_i - s_i + 1$, *is included by a node at level i*, where h_i is the corresponding window size of level i, and s_i is the shift of level i. This nice property gives us a similar detection algorithm as that for a Shifted Binary Tree. When new data points come, the nodes in a Shifted Aggregation Tree are updated from bottom to up. After a node at level $i+1$ is updated, if the new aggregate is more than $f(h_i - s_i + 2)$, a detailed search is performed on the subwindows of size w, $h_i - s_i + 2 \leq w \leq h_{i+1} - s_{i+1} + 1$. An efficient Shifted Aggregation Tree should balance between the updating time and the detailed searching time.

3. Heuristic state-space algorithm to search an efficient Shifted Aggregation Tree

In order to find an efficient Shifted Aggregation Tree (SAT), the classical AI state-space search algorithm is used, given the sample input series and window thresholds. Each SAT is seen as a state. By adding a level onto the top of SAT B, if we can get another SAT A, we say state B can be transformed to state A. The algorithm starts from a SAT having level 0 only, then keeps transforming the candidate set of SATs, until a set of final SATs (i.e. those which can detect bursts in all windows of interst) are reached. In order to find the best SAT, the best-first strategy is used to explore the state space. Each state is associated with a cost. The state with the minimum cost is picked as the next state to be explored, and the final SAT with the minimum cost is picked as the desired structure.

The cost associated with each state is used to indicate which structure to choose in term of running time. We use a theoretical cost model – the expected number of operations – to model the CPU running time. Our model is a simple RAM model: all operations (updates and comparisons) take constant time. The total cost is the sum of the number of updating operations and the expected number of comparison operations, given the sample input series, the window thresholds and the tree structure. Our experiment shows that the theoretical cost model models the actual CPU running time well [10].

4. Empirical Results

4.1. Synthetic Data

Two classes of probabilistic distributions widely used to model many real world applications were chosen to generate the synthetic data: the Poisson distribution and the exponential distribution. For each distribution, we synthesized a set of data with different parameters in a broad range. We want to see how different distributions and thresholds affect the desired Shifted Aggregation Trees.

Assume that each point in the input time series has a mean μ and a standard deviation σ. Assume that for each window size, the probability to exceed the threshold be some value p. We can infer [10] that the alarm probability P_a for an aggregate of window size W to exceed the thresold for window size w is

$$\Phi((\sqrt{T} - \frac{1}{\sqrt{T}})\sqrt{w}\frac{\mu}{\sigma} + \frac{\Phi^{-1}(p)}{\sqrt{T}})$$

where $T = W/w$, denoted the *bounding ratio*, and $\Phi(x)$ is the normal cumulative distribution function.

So P_a is determined by the distribution parameters μ and σ, the threshold parameter p, the bounding ratio T and the level w in the underlying aggregation pyramid. The experiments on both Poisson data and exponential data also show the following:

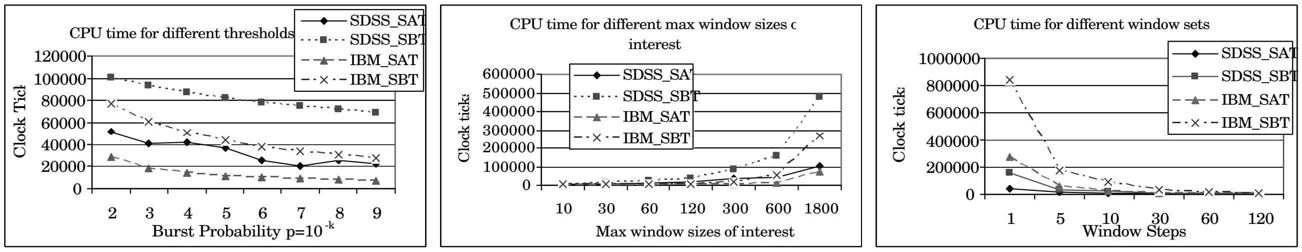
- The larger the ratio $\frac{\mu}{\sigma}$ is, the larger the alarm probability P_a. To mitigate this, the Shifted Aggregation Tree becomes denser in order to bring down P_a. When $\frac{\mu}{\sigma}$ becomes very large, P_a is close to 1. So the structure turns sparser again to reduce the updating time, but is essentially useless.
- The smaller the burst probability p, the larger the threshold, the smaller P_a, so the structure becomes sparser since there are less bursts to worry about.
- As the bounding ratio T decreases, so does P_a. In a Shifted Aggregation Tree, T could be very close to 1, e.g. $W = w + 1$, whereas T in a Shifted Binary Tree is designed to be about 4. Thus a SAT is able to adjust its structure to reduce the alarm probability.
- As the size w increases, so does P_a, while the bounding ratio T becomes smaller in the Shifted Aggregation Tree to try to reduce P_a.

In summary, because the Shifted Aggregation Tree can adjust its structure to reduce the alarm probability, it can achieve far better running time than the Shifted Binary Tree.

4.2. Real World Data

We have used two real world data sets to test the proposed framework: the Sloan Digital Sky Survey (SDSS) Website traffic data which records all the traffic to the SDSS website in 2003, and the NYSE TAQ stock data which includes tick-by-tick trading activities of the IBM stock between Jan. 1st, 2001 to May 31st, 2004. All the events within the same second are aggregated as a single datum. The statistics show that the SDSS data follows the Poisson distribution while the IBM data is close to the exponential distribution.

We are interested in comparing the Shifted Aggregation Tree (SAT) with the Shifted Binary Tree (SBT) under different settings.



(a) Thresholds (b) Maximum window size of interest (c) Different set of window sizes of interest

Figure 2. Performance test: CPU time comparison on the Sloan Digital Sky Survey (SDSS) weblog data and the IBM stock data

- Different thresholds
  The thresholds are set to reflect a burst probability ranging from 10^{-2} to 10^{-9}. Figure 2.a shows the results for both data sets. As the burst probability decreases, the CPU time for the SAT decreases quickly.
- Different maximum window sizes of interest
  The maximum window sizes are set from 10 seconds up to 1800 seconds. Figure 2.b shows as the maximum window size increases, there are more possible levels to adjust the bounding ratio, thus the speedup for the SAT over the SBT increases.
- Different sets of window sizes
  Instead of detecting bursts at every window size, we want to detect bursts for a set of window sizes $n, 2*n, 3*n, ...$, where n is set to be 1, 5, 10, 30, 60, 120 respectively. Figure 2.c shows that as the set of window sizes becomes sparser, there are fewer bursts to worry about, thus both the SBT and the SAT take less time.

We also studied how sensitive the desired Shifted Aggregation Tree is to the training data and how the search parameters in the state-space algorithm affect the desired structure. The results can be found in [10].

5. Related Work

As an interesting and important phenomenon, monitoring and mining bursty behaviors are attracting more and more interests recently.

Wang et al. [9] models the bursty behavior in self-similar time series, such as Ethernet, disk traffic, etc. Kleinberg [3] studies the bursty and hierarchical structure in temporal text stream to discover how high frequency words change over time. Their work focus on modeling the bursty behaviors, while our focus is a high-performance algorithm to detect bursts, thus complements their work. Vlachos et al. [8] mine the bursty behavior in the query logs of the MSN search engine. We share the view that burst detection should be a preliminary primitive for further knowledge mining process.

Neill et al. [5, 6, 4] study the problem of detecting significant spatial clusters in multidimensional space. They consider a general non-monotonic density function and use the overlap-kd tree with a fixed overlapping structure independent of the input data. Our technique can adapt to different characteristics of the input data and be applied to their application to achieve better performance.

Datar et al. and Gibbons et al. [1, 2] study a related problem: estimating the number of 1's in a 0-1 stream and the sum of bounded integers in an integer stream in the last N elements. They use synopsis structures called Exponential Histograms and Waves respectively. Like our Shifted Aggregation Tree, these are multiresolution aggregation structures, though with coarser aggregation levels for the past and finer aggregation levels for recent data.

6. Conclusion

In this paper, we propose a framework for adaptive and therefore better elastic burst detection. It includes a family of data structures and a heuristic search algorithm to find an efficient structure given the input. Experiments show an improvement factor of up to 35 times depending on the input.

References

[1] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM*, 31(6), September 2002.

[2] P.B. Gibbons and S. Tirthapura. Distributed stream algorithms for sliding windows. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 63–72, 2002.

[3] J. Kleinberg. Bursty and hierarchical structure in streams. In *KDD '02: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 91–101, New York, NY, USA, 2002. ACM Press.

[4] D.B. Neill and A. Moore. Rapid detection of significant spatial clusters. Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004.

[5] D.B. Neill and A.W. Moore. A fast multi-resolution method for detection of significant spatial disease clusters. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[6] D.B. Neill, A.W. Moore, F. Pereira, and T. Mitchell. Detecting significant multidimensional spatial clusters. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 969–976, Cambridge, MA, 2005. MIT Press.

[7] D. Shasha and Y. Zhu. *High Performance Discovery in Time Series: Techniques and Case Studies*. Springer, 2003.

[8] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 131–142, New York, NY, USA, 2004. ACM Press.

[9] M. Wang, T. Madhyastha, N.H. Chan, S. Papadimitriou, and C. Faloutos. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, pages 507–516, Washington, DC, USA, 2002. IEEE Computer Society.

[10] X. Zhang and D. Shasha. High performance burst detection. *Technical Report, New York University*, 2005.