Volume 3, No. 5, Sept-Oct 2012
ISSN No. 0976-5697
International Journal Of Advanced Research In Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Parallel Anchors Hierarchy to Document Signatures Using Sparse Spatial Selection

Afrin Md*
Department of Computer Science and Engineering
School of Information Technology, JNTUH
Hyderabad, A.P, India
affu_afrin@yahoo.com

Sheena Mohammed
Department of Computer Science and Engineering
Vardhaman College of Engineering
Hyderabad, A.P, India
sheenamd786@gmail.com

Abstract- Most of the automated documents clustering algorithms face the challenges of high cost and performance in comparing documents in large text corpora. Parallel anchors hierarchy is an algorithm that localizes data based on triangle inequality obeying distance metric, the algorithm strives to minimize the number of distance calculations needed to cluster the documents into "anchors" around reference documents called "pivots" and also improves the performance of clustering by adding parallelism to it. This algorithm selects the initial pivot at random and also decides the number of anchors to create before anchors formation. This may affect the clustering quality. In this paper, we extend the original parallel anchors hierarchy algorithm to improve the clustering quality and performance by adding dynamic pivot selection technique which decides the number of anchors to create based on the dimensionality of the text corpora.

Key words: parallel clustering, anchors hierarchy, Sparse spatial selection, pivot selection, database, documents.

I. INTRODUCTION

Parallel Anchors Hierarchy algorithm starts with the selection of initial pivot at random from the set of documents to form initial anchor. Then, new anchors are selected by choosing furthest e points from any pivot. In next phase, points are associated to new anchors using parallel method which moves points to new anchors simultaneously and associates points to new anchors which are nearer to them. The new anchors to which points are associated are stored in storage X. This storage X is used to move points to their nearest anchors simultaneously. This algorithm suits for parallel computers where concurrency is high. In this algorithm as initial anchor is selected at random and as there are no guidelines on selecting the optimal number of anchors, this may affect the quality of the clustering.

To improve the quality of clustering this paper presents a Sparse Spatial Selection (SSS) technique [2] for pivot selection which dynamically selects an efficient set of pivots adapted to the complexity of the database. This technique is presented in Section III.

The rest of the paper is organized as follows: Section II presents with more detail the strategies proposed in [2] and [3], since they are the base of the ones proposed in Section III. Section IV shows and discusses the experimental evaluation and, finally, Section V presents the conclusions of the paper and future work.

II. RELATED WORK

Most of the pivot-based clustering methods choose pivots at random. Furthermore, there are no guidelines to determine the optimal number of pivots, since this parameter depends on the metric space we are working with. In previous work, some heuristics for pivot selection have been proposed. For example, in [8] pivots are objects maximizing the sum of distances between them. [7] and [4] propose heuristics to obtain pivots far away from each other. In [6] the importance of the pivot selection strategy was studied in depth, showing empirically how it affects to the performance of a technique.

The main contribution in [6] is a criterion to compare the efficiency of two sets of pivots of the same size. Let {p1,p2, …, pk} be a set of pivots, with pi ∈ U. Given an object u ∈ U, a tuple that contains all the distances from u to each pivot is denoted as [u] = (d(u,p1), d(u,p2), …, d(u,pk)). Thus there is a space P = {[u] / u ∈ X}, that is also a vector space □k. Over the elements of this space the distance function D{p1, p2, …, pk}([x],[y]) =max{1≤i≤k}| d(x,pi) - d(y, pi)| can be defined. Then, we have a metric space (P, L∞). Under this conditions, given a query q and a radius r, the condition to discard u ∈ U can be seen as |d(pi,u) - d(pi, q)| > r for some pivot pi, in D{p1, p2, …, pk}([q], [u]) > r. Since the more objects a set of pivots can discard, the better it is, then a set of pivots will be better than others if it increases the probability of D{p1, p2, …, pk}([q], [u]) > r. Being μD the mean of the distribution D, that probability increases when μD is maximized. Therefore, authors' criterion establishes that the set of pivots {p1, p2, …, pk} is better than the set of pivots {p1', p2', …, pk'} if μ{p1, p2, …, pk}> μ{p1', p2', , pk'}

In [6] several selection strategies based on the previous efficiency criterion were proposed: i) Random, that chooses the set of pivots randomly; ii) Selection, that selects N random sets of pivots and finally chooses the one maximizing μD; iii) Incremental, in which the next pivot chosen will be that object such that, after adding it to the current set of pivots, maximizes μD; and iv) Local Optimum, an iterative strategy that, starting with a random set of pivots, in each step replaces by a new object, the current pivot which less contributes to μD. The performance obtained by many of these techniques depends on the metric space considered. Their conclusions show that, in general, good pivots should be far from each other and also from the rest of objects in the database.

Unfortunately, this second condition does not always lead to obtain good pivots. Determining the optimal number of pivots (the value k) is an important problem. However, it is known that the efficiency of the searches depends on that

parameter. Moreover, k can vary greatly for different metric spaces. All the pivot selection techniques shown use a pre computed fixed value. In [6] a brute-force approach to determine the optimal number of pivots is used. Results confirm that this number depends greatly on the metric space, and affects the efficiency of the methods. Therefore, adjusting it as well as possible is an interesting problem.

III. PARALLEL ANCHORS HIERARCHY USING SPARSE SPATIAL SELECTION (SSS)

In this section we present Parallel Anchors Hierarchy algorithm using Sparse Spatial Selection (SSS) [2] for selecting optimal number of pivots dynamically. It is an efficient method with some important advantages over the previous work. SSS is a dynamic method since the database can be initially empty and grow/decrease later as objects are inserted/ deleted. As described in [1] Parallel Anchors Hierarchy is preceded by Text Preprocessing and Bag of Words (BoW). Text Preprocessing includes tokenization of strings, stop word removal and stemming. Then BoW creates document signatures (collection of word to-frequency mappings)—a strategy for characterizing documents within a corpus—and to use these signatures as a way to measure similarity between documents. Parallel Anchors Hierarchy performs the clustering of the documents into a navigable structure.

A. Sparse Spatial Selection

Sparse Spatial Selection (SSS) [2] dynamically selects a set of pivots well distributed in the space, and adapted to the complexity of the collection.

Let (X,d) be a metric space, U ⊆ X an object collection, and M the maximum distance between any pair of objects, M = max {d(x,y) / x, y ∈ U}. First, the set of pivots is initialized with the first object of the collection. Then, for each element xi ∈ U, xi is chosen as a new pivot iff its distance to any pivot in the current set of pivots is equal or greater than Mα, being α is a constant which optimal values are around 0.4 (as shown later). That is, an object in the collection becomes a new pivot if it is located at more than a fraction of the maximum distance with respect to all the current pivots. For example, if α = 0.5 an object is chosen as a new pivot if it is located further than a half of the maximum distance from the current pivots.
Algorithm 1 shows the pseudo code of SSS.

Algorithm 1: Sparse Spatial Selection

Input: $(\mathbb{X}, d)$, $\mathbb{U}$, α, M
Output: *Pivots*

$Pivots \leftarrow \{x_1\}$;
foreach $x_i \in \mathbb{U}$ **do**
  if $\forall\, p \in Pivots,\ d(x_i, p) \geq M\alpha$ **then**
    $Pivots \leftarrow Pivots \cup \{x_i\}$;
return *Pivots*

Figure 1 Pseudo code for SSS.

The parameter α directly affects the number of pivots selected by SSS. Experimental results provided in [9], show that the optimal values of this parameter are in the interval [0.35, 0.40] for a wide variety of metric databases, and that the efficiency of SSS is virtually the same for all the values in this interval. The value of the maximum distance M between any pair of objects in the metric space can be estimated from the definition of the space and the distance function, not being necessary to directly compute it. The results presented in [9] show that this strategy is more efficient than others previously proposed in most cases. Furthermore, SSS has other important features. SSS is dynamic, this is, the database can be initially empty, and the pivots will be selected when needed as the database grows. SSS is also adaptive, since it is no necessary to state in advance the number of pivots to select. As the database grows, the algorithm determines if the collection has become complex enough to select more pivots or not. Therefore, SSS adapts the index to the intrinsic dimensionality of the metric space [9]. The selection procedure is also more efficient in SSS than in previous techniques.

B. Parallel Anchors Hierarchy:

The Parallel Anchors Hierarchy using SSS algorithm extends the parallel algorithm [1] by adding anchors dynamically. Original parallel anchors algorithm selects initial pivot randomly, and then multiple anchors are formed by selecting pivots which are far away from the initial pivot. In this paper SSS technique is used to select pivots given in 3.A.

a. Initialization:

The initialization of the parallel algorithm using SSS starts with the selection the first object of the documents collection as initial pivot and computes the distance from every other point to the pivot to form the first anchor.

b. New Anchors Selection:

Instead of selecting furthest e points from any pivot(s), this phase uses SSS for selection of new anchors. The algorithm 1 given above is used to select new anchors. SSS selects a document point as a new pivot if it is far away enough from the pivots already selected. The number of anchors to be selected will be decided by the SSS algorithm.

c. Associating Points with New Anchors:

Parallel algorithm first considers moving each point to every new anchor before it is actually moved to the pivot which is nearest to it. This operation is implemented in two steps: 1. Parallel region to decide where the point should move and 2. Parallel region to actually move the point.

The algorithm used to implement this operation is as follows

Procedure: Associate Points with New Anchors
Input: Anchor of points $\mathbb{C}$
Input: Set of new pivots $\mathcal{A}'$
Input: Number of clusters formed so far, k
Input: Number of new anchors e
Output: Future Anchor associations $\mathbb{X}$

1: $\mathcal{X} \leftarrow$ Inverse index of $\mathbb{C}$
2: $\mathcal{D}_{i \in \{1..k\}} = \|\mathcal{A}_i - \mathcal{P}_{idx(i, j \in \{1..|\mathcal{C}_i|\})}\|$
3: **for all** i in $\{1..k\}$ **do** ▷ Parallel Loop
4: $m \leftarrow \underset{m \in \{1..|\mathcal{D}_i|\}}{\arg\max}\ (\|a_i - \mathcal{D}_{i_m}\|)$
5: **for all** j in $\{1..e\}$ **do** ▷ Parallel Loop
6: $h \leftarrow \frac{1}{2}\|a'_j - a_i\|$
7: **while** $\|a'_j - \mathcal{D}_{i_m}\| < h$ **do**
8: $t \leftarrow \mathcal{X}_{i_m}$
9: **if** $(\|a'_j - \mathcal{D}_{i_m}\| < \|a_t - \mathcal{D}_{i_m}\|)$ **then**
10: $\mathcal{X}_{i_m} \leftarrow j$
11: **end if**
12: $D_i \leftarrow D_i \setminus D_{i_m}$
13: $m \leftarrow \underset{m \in \{1..|\mathcal{D}_i|\}}{\arg\max}\ (\|a_i - \mathcal{D}_{i_m}\|)$
14: **end while**
15: **end for**
16: **end for**

Figure 2 Parallel method for associating points with new anchors

The distance calculations performed from lines 6-9 in above algorithm are used to eliminate the consideration of anchors whose distance from the new pivot are twice more than the radius of that anchor. Then the new anchor's name is stored in X_{im}, which indicates the anchor the point will move at the end of the phase.

As the algorithm performs the operation in parallel loop, the unordered updates may be performed to storage X. So, the updates must be made atomic with respect to other parallel iterations. In multithreading, synchronization can be used to incorporate atomic updates on storage X.

d. Move Points to New Anchors:

The storage X is used to move points to new anchors simultaneously. The algorithm used in moving points to new anchors is given figure 3.

Procedure: Move Points to New Anchors
Input: Anchors $\mathbb{C}$
Input: Set of new pivots $\mathcal{A}'$
Input: Number of clusters formed so far, k
Input: Future Anchor associations $\mathbb{X}$
Output: Anchors $\mathbb{C}$
Output: Set of Pivots $\mathcal{A}$
Output: Number of anchors formed at end of phase, k

1: $\mathbb{C}' \leftarrow \emptyset$
2: **for all** i in $\mathbb{C}$, j in $\mathcal{C}_i$ **do** ▷ Parallel Loop
3: $t \leftarrow \mathcal{X}_{i_j}$
4: **if** $\mathcal{X}_{i_j} \neq i$ **then**
5: $\mathcal{C}'_t \leftarrow \mathcal{C}'_t \cup \mathcal{C}_{i_j}$
6: $\mathcal{C}_i \leftarrow \mathcal{C}_i \setminus \mathcal{C}_{i_j}$
7: **end if**
8: **end for**
9: $\mathbb{C} \leftarrow \mathbb{C} \cup \mathbb{C}'$
10: $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}'$
11: $k \leftarrow k + e$

Figure 3 Parallel method for moving points from old anchors to new anchors and committing new anchors to the anchors list

The loop used to move points to new anchors is implemented as parallel loop to move every point simultaneously. This loop iterates through every anchor and every point in every anchor comparing the new pivot stored in X to the current pivot. If the point is to be moved then it is inserted into the new anchor and removed from the old anchor. After processing all the points, the set of new pivots is merged with the set of old pivots and the set of new anchors is merged with the set of old anchors. Finally, the number of anchors found so far, k, is updated to the current number of anchors

C. Form Hierarchy of Anchors:

The second phase of Parallel Anchors Hierarchy after clustering of points into anchors is hierarchy formation. In this, anchors are combined into a navigable structure. The hierarchy formation process recursively merges two anchors, which when combined forms the smallest new anchor. Then the new anchor replaces the two anchors. The process is repeated until the final hierarchy is formed. The final hierarchy is shown in figure 3.1.4.

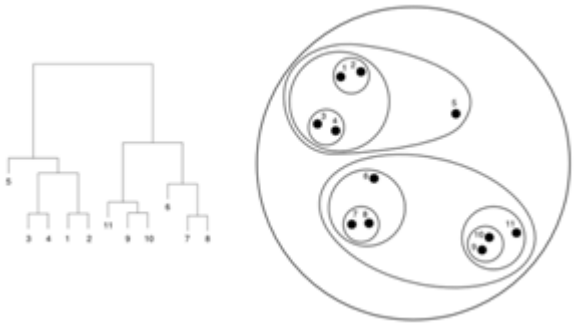

Figure 4 Complete Anchors Hierarchy after multiple rounds of anchor combining

IV. EXPERIMENTAL EVALUATION

A. White Papers:

We downloaded white papers from white paper search engine. A java pdf library was used to extract the page content from documents. However, we are aware that this corpus has some non-words and this is borne out in the data showing so many unique words

We selected only the first 1,000 documents from the over thousands documents. The high variance of document signature (non-zero vector entry) sizes caused load imbalance problems, so we arbitrarily filtered out all documents with fewer than 100 unique words and those documents with more than 5,00 unique words.

B. Performance:

We ran all of the experiments on Ateji PX on multi core processor. Performance is the major reason for embracing parallel programming. A sequential program at full power on a multi-core computer will use only one core, regardless of the available hardware:

Figure 5 CPU usage for sequential program

A parallel program at full power on the same hardware will use all available cores:

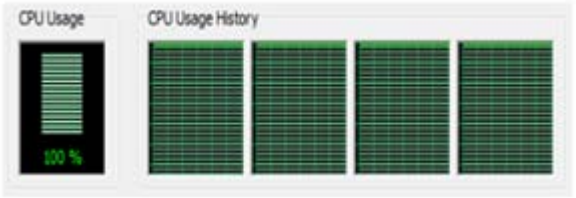

Figure 6 CPU usage for parallel program

But parallelizing programs is far from being a trivial task without appropriate tools. Multi-threaded programs based on thread or task libraries are well known for crashing in unexpected ways, being hard to test and debug, and difficult to evolve.

With Ateji PX, parallelizing an application can be as simple as inserting a "||" operator in the source code.

Ateji PX performs most of its job at compile-time and adds little or no overhead at run-time. The programmer is freed from irrelevant technical details and can more easily concentrate on performance improvement. Experimenting with different parallelization strategies requires little changes in the source code.

We ran our code on the down-selected dataset and observed the speed for computing the anchors; we do not include speed for computing BoW.

Performance/ scalability of Ateji PX on multi core processors is shown in below figure.

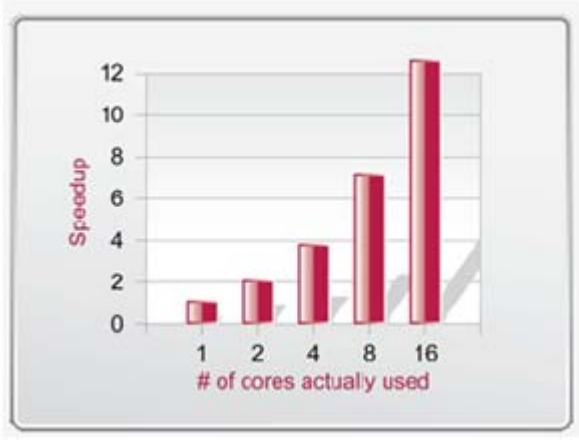

Figure 7 Performance/ scalability on Ateji PX

The number of pivots selected for different values of α for a document set of 1000 is shown in below graph.

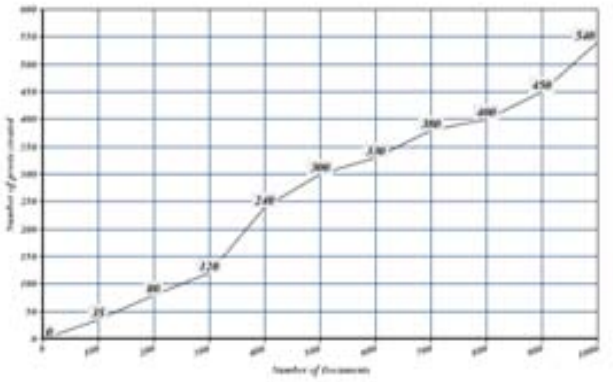

Figure 8 Example of number of pivots selected

In previous method as in [1] the number of pivots to be selected should be decided before the formation of anchors. The technique SSS used in this paper for selection of new pivots is a dynamic method which selects the number of pivots to form based on the value of α and the size of documents database. The above graph is drawn by taking size of documents database on x-axis and number of pivots formed on y-axis at the value of α=0.40. The graph shows how the technique SSS selects the number of pivots dynamically based on the number of document objects. The technique SSS allows database to be empty first and can be expanded by adding new documents.

V. CONCLUSION AND FUTURE WORK

In this paper we propose Parallel Anchors Hierarchy using Sparse Spatial Selection method. The main characteristics of this method are its efficiency and dynamism (which allows the database to be initially empty and grow later). The main contribution of our method is the pivot selection strategy for efficient selection of pivots.

Our experimental results show that the method selects number of pivots that depends on the intrinsic dimensionality of the documents metric space, and not on the number of elements of the collection. In addition, this number of pivots is very similar to the optimum number for other strategies. This makes it unnecessary to state in advance the number of pivots needed for the index structure, something that no method has considered until now. The number of pivots selected is adapted to the space complexity, avoiding the selection of unnecessary pivots which could reduce the search performance. The efficiency of our method in vector spaces is similar to that obtained in previous works. However, our tests show that our method is more efficient than the existing one. Further we will investigate how different pivot selection strategies affect execution performance and cluster quality.

VI. REFERENCES

[1]. "Toward Parallel Document Clustering" Jace A. Mogill, David J. Haglin Pacific Northwest National Laboratory Richland, WA, 99354 USA

[2]. "Spatial Selection of Sparse Pivots for Similarity Search in Metric Spaces" Nieves R. Brisaboa, Antonio Fariña, Óscar Pedreira, Database Laboratory, University of A Coruña

[3]. Tolga Bozkaya and Meral Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In Proceedings of the ACM International Conference on Management of Data (SIGMOD 1997), pages 357-368, May 1997.

[4]. Sergey Brin. "Near neighbor search in large metric spaces". In 21st conference on Very Large Databases (VLDB), 1995.

[5]. "A Dynamic Pivot Selection Technique for Similarity Search", Benjamin Bustos, University of Chile, Oscar Pedreira, Nieves Brisaboa, University of A Coruna

[6]. Benjamín Bustos, Gonzalo Navarro, and Edgar Chávez. Pivot selection techniques for proximity search in metric spaces. In SCCC 2001, Proceedings of the XXI Conference of the Chilean Computer Science Society, pages 33-40. IEEE Computer Science Press, 2001.

[7]. Edgar Chávez, José Luis Marroquín, and Gonzalo Navarro. Overcoming the curse of dimensionality. In European Workshop on Content-based Multimedia Indexing (CBMI'99), pages 57-64, 1999.

[8]. Luisa Micó, José Oncina, and R. Enrique Vidal. A new version of the nearest-neighbor approximating and eliminating search (AESA) with linear pre-processing time and memory requirements. Pattern Recognition Letters, 15:9-17, 1994

[9]. Walter A. Burkhard and Robert M. Keller. Some approaches to best-match file searching. Communications of the ACM, 16(4):230-236, April 1973.